Polynomial Time Algorithms for Branching Markov Decision Processes and Probabilistic Min(Max) Polynomial Bellman Equations

Polynomial Time Algorithms for Branching Markov Decision Processes and Probabilistic Min(Max) Polynomial Bellman Equations*

Kousha Etessami
U. of Edinburgh
kousha@inf.ed.ac.uk

Alistair Stewart
U. of Edinburgh
stewart.al@gmail.com

Mihalis Yannakakis
Columbia U.
mihalis@cs.columbia.edu

Abstract

We show that one can compute the least non-negative solution (a.k.a., *least fixed point*) for a system of probabilistic min (max) polynomial equations, to any desired accuracy $\epsilon > 0$, in time polynomial in both the encoding size of the system and in $\log(1/\epsilon)$.

These are Bellman optimality equations for important classes of *infinite-state* Markov Decision Processes (MDPs), including Branching MDPs (BMDPs), which generalize classic multi-type branching stochastic processes. We thus obtain the first polynomial time algorithm for computing, to any desired precision, optimal (maximum and minimum) *extinction probabilities* for BMDPs. Our algorithms are based on a novel generalization of Newton's method which employs linear programming in each iteration.

We also provide P-time algorithms for computing an ϵ-optimal policy for both maximizing and minimizing extinction probabilities in a BMDP, whereas we note a hardness result for computing an exact optimal policy. Furthermore, improving on prior results, we provide more efficient P-time algorithms for qualitative analysis of BMDPs, i.e., for determining whether the maximum or minimum extinction probability is 1, and, if so, computing a policy that achieves this. We also observe some complexity consequences of our results for branching simple stochastic games, which generalize BMDPs.

1 Introduction

Markov Decision Processes (MDPs) are a fundamental model for stochastic dynamic optimization, with applications in many fields (see, e.g., [29, 20]). They extend purely stochastic processes (Markov chains) with a controller (an agent) who can partially affect the evolution of the process, and seeks to optimize some objective. For many important classes of MDPs, the task of computing the *optimal value* of the objective, starting at any state of the MDP, can be rephrased as the problem of solving the associated *Bellman optimality equations* for that MDP model. In particular, for finite-state MDPs where, e.g., the objective is to maximize (or minimize) the probability of eventually reaching some target state, the associated Bellman equations are *max-(min-)linear* equations, and we know

*A preliminary extended abstract for this paper, [14], appeared in the *Proceedings of the 39th International Colloquium on Automata, Languages, and Programming (ICALP'12)*, 2012 (Fuller preprint on ArXiv:1202.4789). Research partially supported by the Royal Society and by NSF Grants CCF-1320654, CCF-1703925.

how to solve such equations in P-time using linear programming (see, e.g., [29]). The same holds for a number of other classes of finite-state MDPs.

In many important settings however, the state space of the processes of interest, both for purely stochastic processes, as well as for controlled ones (MDPs), is not finite, even though the processes can be specified in a finite way. For example, consider *multi-type branching processes* (BPs) [25, 22], a classic probabilistic model with applications in many areas. A BP models the stochastic evolution of a population of entities of distinct types. In each generation, every entity of each type T produces a set of entities of various types in the next generation according to a given probability distribution on offsprings for the type T. In a *Branching Markov Decision Process* (BMDP) (e.g., [27, 31]), there is a controller who can take actions that affect the probability distribution for the sets of offsprings for each entity of each type.

Branching processes have been used to model phenomena in many fields, including biology (see e.g. [24]), population genetics ([21]), physics and chemistry (e.g. particle systems, nuclear chain reactions), medicine (e.g. cancer growth), marketing, and others. In many cases, the process is not purely stochastic but there is the possibility of taking actions (for example, adjusting the conditions of reactions, applying drug treatments in medicine, advertising in marketing, etc.) which can influence the probabilistic evolution of the process to bias it towards achieving desirable objectives. Branching Markov decision processes are a useful model in such settings. For both BPs and BMDPs, the state space consists of all possible populations, given by the number of entities of the various types, so there are an infinite number of states. From the computational point of view, the usefulness of such infinite-state models hinges on whether their analysis remains tractable.

In recent years there has been a body of research aimed at studying the computational complexity of key analysis problems associated with MDP extensions (and, more general stochastic game extensions) of important classes of finitely-presented but *countably infinite-state* stochastic processes, including controlled extensions of classic multi-type branching processes (i.e., BMDPs), *stochastic context-free grammars*, and discrete-time *quasi-birth-death processes*. In [19] a model called *recursive Markov decision processes* (RMDP) was studied that is in a precise sense more general than all of these, and forms the MDP extension of *recursive Markov chains* [18] (and equivalently, *probabilistic pushdown systems* [11]), or it can be viewed alternatively as the extension of finite-state MDPs with recursion. RMDPs consist of a set of MDPs that can call each other recursively, in the same way as recursive procedures do. These models arise in various areas, e.g. performance evaluation, computational biology, and program analysis and verification.

A central analysis problem for all of these models, which forms the key to a number of other analyses, is the problem of computing their *optimal termination (extinction) probability*. For example, in the setting of multi-type Branching MDPs (BMDPs), these key quantities are the maximum (minimum) probabilities, over all control strategies (or policies), that starting from a single entity of a given type, the process will eventually reach extinction (i.e., the state where no entities have survived). From these quantities, one can compute the optimum probability for any initial population, as well as other quantities of interest.

One can indeed form Bellman optimality equations for the optimal extinction probabilities of BMDPs, and for a number of related important infinite-state MDP models. However, it turns out that these optimality equations are no longer max/min *linear* but rather are max/min *polynomial* equations ([19]). Specifically, the Bellman equations for BMDPs with the objective of maximizing (or minimizing) extinction probability are multivariate systems of monotone probabilistic max (or min) polynomial equations, which we call **max/minPPSs**, of the form $x_i = P_i(x_1, \ldots, x_n)$, $i = 1, \ldots, n$,

where each $P_i(x) \equiv \max_j q_{i,j}(x)$ (respectively $P_i(x) \equiv \min_j q_{i,j}(x)$) is the max (min) over a finite number of probabilistic polynomials, $q_{i,j}(x)$. A *probabilistic polynomial*, $q(x)$, is a multi-variate polynomial where the monomial coefficients and constant term of $q(x)$ are all non-negative and sum to ≤ 1. We write these equations in vector form as $x = P(x)$. Then $P(x)$ defines a mapping $P : [0,1]^n \to [0,1]^n$ that is monotone, and thus (by Tarski's theorem) has a *least fixed point* in $[0,1]^n$. The equations $x = P(x)$, can have more than one solution, but it turns out that the optimal value vector for extinction probabilities in the corresponding BMDP is precisely the *least fixed point* (LFP) solution vector $q^* \in [0,1]^n$, i.e., the (coordinate-wise) least non-negative solution ([19]). Intuitively, the reason that the optimal extinction probabilities are given by the least fixed point (as opposed to any other fixed point) is the following. For any integer $t \geq 0$, let $p(t)$ denote the vector of optimal probabilities of extinction by time t. Clearly, $p(t)$ converges monotonically from below to the optimal extinction probabilities as t goes to infinity, meaning $\lim_{t\to\infty} p(t)$ is the vector of optimal extinction probabilities. For a max/minPPS $x = P(x)$, and for an integer $t \geq 0$, define the vector $P^t(0) \in [0,1]^n$ inductively by $P^0(0) := 0$, and $P^{t+1}(0) := P(P^t(0))$. It is not hard to show that $P^t(0) = p(t)$ for all t. Due to the monotonicity of $P(\cdot)$, the LFP solution of $x = P(x)$ is given by $\lim_{t\to\infty} P^t(0)$. Thus $\lim_{t\to\infty} P^t(0) = q^* = \lim_{t\to\infty} p(t)$, so the LFP is indeed the vector of optimal extinction probabilities.

The same class of equations (max/minPPS) also models other stochastic processes besides BMDPs, including, controlled stochastic context-free grammars, and 1-exit Recursive Markov Decision Processes (1-RMDP), i.e. RMDPs where the component MDPs have one exit (terminating state). These models arise in various areas. For example, RMDPs are a natural model for recursive probabilistic programs, where the component MDPs of the RMDP correspond to the procedures of the program. There has been an extensive body of work over many years that has developed the theory, algorithms, and tools for the analysis and verification of non-recursive probabilistic programs, which are modeled abstractly by ordinary finite-state Markov decision processes. Extending the scope of the work to handle recursive probabilistic programs requires the analysis of RMDPs. A central problem for this is the termination problem, computing the worst-case (or best-case) probability of termination of the RMDP; this is essential for the analysis and verification of more complex temporal properties. The termination probabilities of 1-exit RMDPs can be captured by max/minPPS, i.e. for every 1-RMDP one can construct efficiently a maxPPS (or minPPS), whose LFP gives the maximum (or minimum) termination probability of the 1-RMDP.

Already for pure stochastic multi-type branching processes (BPs), the extinction probabilities may be irrational values. The problem of *deciding* whether the extinction probability of a BP is $\geq p$, for a given probability p is in PSPACE ([18]), and likewise, deciding whether the optimal extinction probability of a BMDP is $\geq p$ is in PSPACE ([19]). These PSPACE upper bounds appeal to decision procedures for the existential theory of reals for solving the associated (max/min)PPS equations. However, already for BPs, it was shown in [18] that this quantitative *decision* problem is already at least as hard as the *square-root sum* (`Sqrt-Sum`) problem, as well as a (much) harder and more fundamental problem called `PosSLP`, which captures the power of unit-cost exact rational arithmetic. It is a long-standing open problem whether either of these decision problems is in NP, or even in the polynomial time hierarchy (see [1, 18] for more information on these problems). Thus, such *quantitative decision problems* are unlikely to have P-time algorithms in the standard Turing model, even in the purely stochastic setting, so we can certainly not expect to find P-time algorithms for the extension of these models to the MDP setting.[1] On the other hand, it was shown in [18] and

[1]Let us mention however that, more recently in [13] we have also shown that the quantitative decision problem

[19], that for both BPs and BMDPs the *qualitative* decision problem of deciding whether the optimal extinction probability $q_i^* = 0$ or whether $q_i^* = 1$, can be solved in polynomial time.

The hardness of the quantitative decision problem does not however rule out the possibility of efficiently approximating the optimal extinction probabilities to any desired precision. A simple approach for approximation is to apply *Value Iteration*: Starting with $x = 0$, repeatedly apply P, to compute $P(0), P^2(0), \ldots, P^k(0), \ldots$. As $k \to \infty$, $P^k(0)$ converges (monotonically) to the LFP q^*. However, the convergence can be very slow, even for some simple examples of pure branching processes, specifically it can be double-exponential both in the number of types and in the number of bits of precision. The extinction probabilities of pure BPs are the LFP of a system of probabilistic polynomial equations (PPS), without max or min. Consider the equation $x = 0.5x^2 + 0.5$, which corresponds to a simple BP with one type; the LFP is 1 but, as shown in [18], Value Iteration needs 2^{k-3} iterations to approximate it with k bits of precision. Furthermore, there are pure multi-type BPs, based on "nesting" this example, namely the system with $n+1$ variables given by $x_0 = 0.5x_0^2 + 0.5$ and $x_i = 0.5x_i^2 + 0.5x_{i-1}$, for $i = 1, \ldots, n$, where approximating the extinction probability within less than $1/2$ (i.e. getting one bit of precision) using value iteration, starting from the 0 vector, requires double-exponential number of iterations in n, specifically at least 2^{2^n-3} iterations (see, [10, 30]); for example, if $n = 10$, the value of x_{10} remains close to 0 (i.e. very far from the LFP value 1) even after 2^{1000} iterations. It is also known that for ordinary finite-state MDPs, value iteration (as well as policy iteration) requires in some cases an exponential number of iterations.

Despite decades of theoretical and practical work on computational problems like extinction relating to multi-type branching processes, and equivalent termination problems related to stochastic context-free grammars, until recently it was not even known whether one could obtain *any* non-trivial *approximation* of the extinction probability of a purely stochastic multi-type branching processes (BP) in P-time. In recent work [12, 13], we provided the first polynomial time algorithm for computing (i.e., approximating) to within any desired additive error $\epsilon > 0$ the LFP of a given PPS, and hence the extinction probability vector q^* for a given purely stochastic BP, in time polynomial in both the encoding size of the PPS (or the BP) and in $\log(1/\epsilon)$. The algorithm works in the standard Turing model of computation. Our algorithm was based on an approach using Newton's method that was first introduced and studied in [18]. In [18] the approach was studied for more general systems of *monotone* polynomial equations (MPSs), and it was subsequently further studied in [10]. The algorithm of [12, 13] for PPS first identifies and removes the variables that have value 0 or 1 in the LFP and then applies Newton's method in the resulting system (the removal of the variables with value 0 or 1 is critical for the correctness and efficiency of the algorithm).

Note that unlike PPSs and MPSs, the min/maxPPSs that define the Bellman equations for BMDPs are no longer differentiable functions (they are only piecewise differentiable). Thus, a priori, it is not even clear how one could apply a Newton-type method toward solving them.

Our Results

(1) In this paper we provide the first polynomial time algorithms for approximating the LFP of both maxPPSs and minPPSs, and thus the first polynomial time algorithm for computing (to within any desired additive error) the optimal value vector for BMDPs with the objective of maximizing or minimizing their extinction probability.

Our approach is based on a *generalized Newton's method* (GNM), that extends Newton's method in a natural way to the (non-differentiable) setting of max/minPPSs. The method is again iterative

for the extinction probability of BPs, and for the LFP of PPSs, is in fact polynomial-time equivalent to `PosSLP`.

where each iteration involves the computation of the least (greatest) solution of a max- (min-) linear system of equations, both of which we show can be solved using linear programming. Our algorithms based on GNM have the nice feature that they are relatively simple, although the analysis of their correctness and time complexity is rather involved. We show that, if we first identify and remove the variables that have value 0 or 1 in the LFP, and then apply GNM with a suitable rounding of the computed points along the way, the algorithm computes the LFP of a maxPPS or minPPS within any desired number j of bits of precision (i.e., within additive error 2^{-j}) in polynomial time in the encoding size of the system and the number j of bits of precision. We note that the two cases of maxPPS and minPPS are not symmetric, and separate analysis is required for them (and this holds also for the other results below). The reason for the asymmetry is that we seek a specific fixed point, the least one, and this behaves differently with respect to max and min.

(2) We furthermore show that we can compute ϵ-optimal (pure) strategies (policies) for both maxPPSs and minPPSs (and max/min BMDPs), for *any* given desired $\epsilon > 0$, in time polynomial in both the encoding size of the max/minPPS and in $\log(1/\epsilon)$. This result is at first glance rather surprising, because there are only a bounded number of distinct pure policies for a max/minPPS, and computing an optimal policy is `PosSLP`-hard, as we show, thus it is very unlikely that an optimal policy can be computed in P-time [12].

(3) We provide new algorithms for the *qualitative* analysis of max/minPPS and BMDPs (i.e., identifying the variables that have value 0 or 1 in the LFP), which improve significantly on the running time of the previous P-time qualitative algorithms given in [19]. This is important for the practical efficiency of our quantitative algorithms for the approximation of the LFP, which make crucial use of a preprocessing step that identifies and removes the variables with value 0 or 1 in the LFP. Polynomial time algorithms for the qualitative analysis were first established in [19], but the running time was rather high, especially in the case of maxPPS, which involved the solution of linear programs (LPs) with a cubic number of variables. This is improved substantially in our new qualitative algorithms which solve LPs with a linear number of variables and constraints.

(4) Finally, we consider *Branching simple stochastic games* (BSSGs), which are two-player turn-based stochastic games, where one player wants to maximize, and the other wants to minimize, the extinction probability. The *value* of these games (which are determined) is characterized by the LFP solution of associated min-maxPPSs which combine both min and max operators (see [19]). We observe that our results easily imply a TFNP upper bound for ϵ-approximating the *value* of BSSGs and computing ϵ-optimal strategies for them.

Related work: We have already mentioned some of the important relevant results. BMDPs and related processes have been studied previously in both the operations research (e.g. [27, 31, 8]) and computer science literature (e.g. [19, 9, 5]), but no efficient algorithms were known for the (approximate) computation of the relevant optimal probabilities and policies; the best known upper bound was PSPACE [19].

In [19] we introduced Recursive Markov Decision Processes (RMDPs), a recursive extension of MDPs. We showed that for general RMDPs, the problem of computing the optimal termination probabilities, even within any nontrivial approximation, is undecidable. However, we showed for the important class of 1-exit RMDPs (1-RMDP), the optimal probabilities can be expressed by min (or max) PPSs, and in fact the problems of computing (approximately) the LFP of a min/maxPPS and the termination probabilities of a max/min 1-RMDP, or BMDP, are all polynomially equivalent. We furthermore showed in [19] that there are always pure, memoryless (and "stackless") optimal policies for both maximizing and minimizing termination probability for 1-RMDPs, and likewise

pure memoryless "static" optimal policies for extinction probabilities of BMDPs, as well as for the more general turn-based simple stochastic games (1-RSSGs and BSSGs) which generalize 1-RMDPs and BMDPs.

In [16], 1-RMDPs (and 1-RSSGs) with a different objective were studied, namely optimizing the total expected reward in a setting with positive rewards. In that setting, things are much simpler: the Bellman equations turn out to be max/min-linear, the optimal values are rational, and they can be computed *exactly* in P-time using linear programming.

A work that is more closely related to this paper is [9] by Esparza, Gawlitza, Kiefer, and Seidl. They studied more general monotone min-maxMPSs, i.e., systems of monotone polynomial equations that include both min and max operators, and they presented two different iterative analogs of Newton's methods for approximating the LFP of a min-maxMPS, $x = P(x)$. Their methods are related to ours, but differ in key respects. Both of their methods use certain piecewise linear functions to approximate the min-maxMPS in each iteration, which is also what one does to solve each iteration of our generalized Newton's method. However, the precise nature of their piecewise linearizations, as well as how they solve them, differ in important ways from ours, even when they are applied in the specific context of maxPPSs or minPPSs. They showed, working in the unit-cost *exact* arithmetic model, that using their methods one can compute j "valid bits" of the LFP (i.e., compute the LFP within relative error at most 2^{-j}) in $k_P + c_P \cdot j$ iterations, where k_P and c_P are terms that depend in *some* way on the input system, $x = P(x)$. However, they give no upper bounds on k_P, and their upper bounds on c_P are exponential in the number n of variables of $x = P(x)$. Note that MPSs are more difficult to solve: even without the min and max operators, we know that it is `PosSLP`-hard to approximate their LFP within any nontrivial constant additive error $c < 1/2$, even for pure MPSs that arise from Recursive Markov Chains [18].

Another subclass of RMDPs, called *one-counter MDPs* (a controlled extension of one-counter Markov chains and Quasi-Birth-Death processes [17]) has been studied, and the approximation of their optimal termination probabilities was recently shown to be computable, but only in *exponential time* ([4]). This subclass is incomparable with 1-RMDPs and BMDPs, and does not have min/maxPPSs as Bellman equations.

Organization of the paper: Section 2 gives formal definitions and background on branching Markov decision processes, and max and min probabilistic polynomial systems. In Section 3, we define the Generalized Newton method for maxPPS and minPPS, we analyze the method, and show how to compute the LFP of a maxPPS or a minPPS to desired precision in polynomial time in the encoding size of the system and the desired number of bits of precision. In Section 4 we observe that computing an optimal policy is `PosSLP`-hard (thus probably intractable), and show how to compute an ϵ-optimal policy (for any given $\epsilon > 0$) of a maxPPS or minPPS in polynomial time in the size of the system and $\log(1/\epsilon)$. In Section 5 we show that the approximate computation problems of the value of, and ϵ-optimal strategies for, Branching Simple Stochastic Games (BSSGs) are in TFNP. Our improved polynomial-time algorithms for the qualitative analysis of maxPPSs and minPPSs (and BMDPs) are presented in Appendix A. Proofs for some of the supporting lemmas from the main body are deferred to Appendix B.

2 Definitions and Background

Throughout this paper, it will be convenient to compare a vector or matrix to the all 0, or all 1, vector/matrix. For a given vector/matrix z, we will use the notation $z \geq 0$, $z < 1$, ...,

to denote that every entry of z is, respectively, ≥ 0, < 1, ... The l_∞ norm of a vector z is $\|z\|_\infty := \max_i |z_i|$, and its associated matrix norm $\|A\|_\infty$ is the maximum absolute-value row sum of A, i.e., $\|A\|_\infty := \max_i \sum_j |A_{i,j}|$.

For an n-vector of variables $x = (x_1, \ldots, x_n)$, and a vector $v \in \mathbb{N}^n$, we use the shorthand notation x^v to denote the monomial $x_1^{v_1} \ldots x_n^{v_n}$. Let $\langle \alpha_r \in \mathbb{N}^n \mid r \in R \rangle$ be a multi-set of n-vectors of natural numbers, indexed by the set R. Consider a multi-variate polynomial $P_i(x) = \sum_{r \in R} p_r x^{\alpha_r}$, for some rational-valued coefficients p_r, $r \in R$. We shall call $P_i(x)$ a ***monotone polynomial*** if $p_r \geq 0$ for all $r \in R$. If in addition, we also have $\sum_{r \in R} p_r \leq 1$, then we shall call $P_i(x)$ a ***probabilistic polynomial***.

Definition 2.1. *A* **probabilistic** *(respectively,* **monotone***)* **polynomial system of equations**, $x = P(x)$, *which we shall call a* **PPS** *(respectively, a* **MPS***), is a system of n equations, $x_i = P_i(x)$, in n variables $x = (x_1, x_2, ..., x_n)$, where for all $i \in \{1, 2, ...n\}$, $P_i(x)$ is a probabilistic (respectively, monotone) polynomial.*

A **maximum-minimum probabilistic polynomial system of equations**, $x = P(x)$, *called a* **max-minPPS** *is a system of n equations in n variables $x = (x_1, x_2, \ldots, x_n)$, where for all $i \in \{1, 2, \ldots, n\}$, either:*

- `Max-polynomial`*: $P_i(x) = \max\{q_{i,j}(x) : j \in \{1, ..., m_i\}\}$, Or:*
- `Min-polynomial`*: $P_i(x) = \min\{q_{i,j}(x) : j \in \{1, ..., m_i\}\}$*

where each $q_{i,j}(x)$ is a probabilistic polynomial, for every $j \in \{1, \ldots, m_i\}$.

We shall call such a system a **maxPPS** *(respectively, a* **minPPS***) if for every $i \in \{1, \ldots, n\}$, $P_i(x)$ is a* `Max-polynomial` *(respectively, a* `Min-polynomial`*).*

Note that we can view a PPS in n variables as a maxPPS, or as a minPPS, where $m_i = 1$ for every $i \in \{1, \ldots, n\}$.

For computational purposes we assume that all the coefficients are rational. We assume that the polynomials in a system are given in sparse form, i.e., by listing only the nonzero monomial terms, with the coefficient and the nonzero exponents of each variable in the term given in binary. We let $|P|$ denote the total bit encoding length of a system $x = P(x)$ under this representation.

We use **max/minPPS** to refer to a system of equations, $x = P(x)$, that is either a maxPPS or a minPPS. While [19] also considered systems of equations containing both max and min equations (which we refer to as **max-minPPSs**), our primary focus will be on systems that contain just one or the other. (But we shall also obtain results about max-minPPSs as a corollary.)

As was shown in [19], any max-minPPS, $x = P(x)$, has a **least fixed point** (**LFP**) solution, $q^* \in [0, 1]^n$, i.e., $q^* = P(q^*)$ and if $q = P(q)$ for some $q \in [0, 1]^n$ then $q^* \leq q$ (coordinate-wise inequality). As observed in [18, 19], q^* may in general contain irrational values, even in the case of PPSs. The central results of this paper yield P-time algorithms for computing q^* to within arbitrary precision, both in the case of maxPPSs and minPPSs. As we shall explain, our P-time upper bounds for computing (to within any desired accuracy) the least fixed point of maxPPSs and minPPSs will also yield, as corollaries, FNP upper bounds for computing approximately the LFP of max-minPPSs.

Definition 2.2. *We define a* **policy** *for a max/minPPS, $x = P(x)$, to be a function $\sigma : \{1, ...n\} \to \mathbb{N}$ such that $1 \leq \sigma(i) \leq m_i$.*

Intuitively, for each variable, x_i, a policy selects one of the probabilistic polynomials, $q_{i,\sigma(i)}(x)$, that appear on the RHS of the equation $x_i = P_i(x)$, and which $P_i(x)$ is the maximum/minimum over.

Definition 2.3. *Given a max/minPPS $x = P(x)$ over n variables, and a policy σ for $x = P(x)$, we define the PPS $x = P_\sigma(x)$ by:*

$$(P_\sigma)_i(x) = q_{i,\sigma(i)}(x)$$

for all $i \in \{1, \ldots, n\}$.

Obviously, since a PPS is a special case of a max/minPPS, every PPS also has a unique LFP solution (this was established earlier in [18]). Given a max/minPPS, $x = P(x)$, and a policy, σ, we use q^*_σ to denote the LFP solution vector for the PPS $x = P_\sigma(x)$.

Definition 2.4. *For a maxPPS, $x = P(x)$, a policy σ^* is called* **optimal** *if for all other policies σ, $q^*_{\sigma^*} \geq q^*_\sigma$. For a minPPS $x = P(x)$ a policy σ^* is optimal if for all other policies σ, $q^*_{\sigma^*} \leq q^*_\sigma$. A policy σ is* **ϵ-optimal** *for $\epsilon > 0$ if $||q^*_\sigma - q^*||_\infty \leq \epsilon$.*

A non-trivial theorem, established in [19], is that optimal policies always exist, and furthermore that they actually attain the LFP q^* of the max/minPPS:

Theorem 2.5 ([19], Theorem 2)**.** *For any max/minPPS, $x = P(x)$, there always exists an optimal policy σ^*, and furthermore $q^* = q^*_{\sigma^*}$.*[2]

Probabilistic polynomial systems can be used to capture central probabilities of interest for several basic stochastic models, including Multi-type Branching Processes (BP), Stochastic Context-Free Grammars (SCFG) and the class of 1-exit Recursive Markov Chains (1-RMC) [18]. Max- and minPPSs can be similarly used to capture the central optimum probabilities of corresponding stochastic optimization models: (Multi-type) Branching Markov Decision processes (BMDP), Context-Free MDPs (CF-MDP), and 1-exit Recursive Markov Decision Processes (1-RMDP) [19]. We now define BMDPs and 1-RMDPs.

A **Branching Markov Decision Process** (BMDP) consists of a finite set $V = \{T_1, \ldots, T_n\}$ of types, a finite set A_i of actions for each type, and a finite set $R(T_i, a)$ of probabilistic rules for each type T_i and action $a \in A_i$. Each rule $r \in R(T_i, a)$ has the form $T_i \xrightarrow{p_r} \alpha_r$, where α_r is a finite multi-set whose elements are in V, $p_r \in (0, 1]$ is the probability of the rule, and the sum of the probabilities of all the rules in $R(T_i, a)$ is equal to 1: $\sum_{r \in R(T_i,a)} p_r = 1$.

Intuitively, a BMDP describes the stochastic evolution of entities of given types in the presence of a controller that can influence the evolution. A population X is a finite set of entities of given types, and it can be represented by a vector $v(X) \in \mathbb{N}^n$, where $v_i(X)$ is the number of entities of X of type T_i. Starting from an initial population X_0 at time (generation) 0, a sequence of populations $X_1, X_2, \ldots$ is generated, where X_k is obtained from X_{k-1} as follows. First the controller selects for each entity of X_{k-1} an available action for the type of the entity; then a rule is chosen independently and simultaneously for every entity of X_{k-1} probabilistically according to the probabilities of the rules for the type of the entity and the selected action, and the entity is replaced by a new set of entities with the types specified by the right-hand side of the rule. The process is repeated as long

[2]Theorem 2 of [19] is stated in the more general context of 1-exit Recursive Simple Stochastic Games and shows that also for max-minPPSs, both the max player and the min player have optimal policies that attain the LFP q^*.

as the current population X_k is nonempty, and terminates if and when X_k becomes empty. The objective of the controller is either to minimize the probability of termination (i.e., extinction of the population), in which case the process is a minBMDP, or to maximize the termination probability, in which case it is a maxBMDP. At each stage, k, the controller is allowed in principle to select the actions for the entities of X_k based on the whole past history, may use randomization (a mixed strategy) and may make different choices for entities of the same type. However, it turns out that these flexibilities do not increase the controller's power, and there is always an optimal pure, memoryless strategy that always chooses the same action for all entities of the same type ([19]).

For each type T_i of a minBMDP (respectively, maxBMDP), let q_i^* be the minimum (resp. maximum) probability of termination if the initial population consists of a single entity of type T_i. From the given minBMDP (maxBMDP) we can construct a minPPS (resp. maxPPS) $x = P(x)$ whose LFP is precisely the vector q^* of optimal termination (extinction) probabilities (see Theorem 20 of [19]): The min/max polynomial $P_i(x)$ for each type T_i contains one polynomial $q_{i,j}(x)$ for each action $j \in A_i$, with $q_{i,j}(x) = \sum_{r \in R(T_i,j)} p_r x^{\alpha_r}$.

Example 2.1. Suppose there are two types of entities (for example, bacteria), T_1, T_2. For type T_1 we have three available actions a_1, a_2, a_3. Under a_1, a type-T_1 entity dies with probability 0.3 and produces two T_1 offspring with probability 0.7. We can write these rules succinctly as $R(T_1, a_1) = \{T_1 \stackrel{0.3}{\to} \emptyset, T_1 \stackrel{0.7}{\to} T_1T_1\}$, where T_1T_1 denotes the multi-set $\{T_1, T_1\}$. Action a_2 increases the probability of death to 0.4 but also introduces the possibility that one of the offspring is mutated to the more resilient type T_2 with probability 0.1, i.e., $R(T_1, a_2) = \{T_1 \stackrel{0.4}{\to} \emptyset,\ T_1 \stackrel{0.1}{\to} T_1T_2,\ T_1 \stackrel{0.5}{\to} T_1T_1\}$. Action a_3 has rules $R(T_1, a_3) = \{T_1 \stackrel{0.5}{\to} \emptyset,\ T_1 \stackrel{0.3}{\to} T_1T_2,\ T_1 \stackrel{0.2}{\to} T_1T_1\}$. For type T_2 we have two actions available b_1, b_2. The rules are $R(T_2, b_1) = \{T_2 \stackrel{0.2}{\to} \emptyset,\ T_2 \stackrel{0.5}{\to} T_1T_2,\ T_2 \stackrel{0.3}{\to} T_2T_2\}$, and $R(T_2, b_2) = \{T_2 \stackrel{0.3}{\to} \emptyset,\ T_2 \stackrel{0.2}{\to} T_1T_2,\ T_2 \stackrel{0.5}{\to} T_2T_2\}$. The goal is to choose a strategy that maximizes the probability of extinction.

The corresponding maxPPS has two variables x_1, x_2 for the optimal extinction probabilities of the two types T_1, T_2, and has equations $x_1 = \max\{0.7x_1^2+0.3, 0.5x_1^2+0.1x_1x_2+0.4, 0.2x_1^2+0.3x_1x_2+0.5\}$, and $x_2 = \max\{0.5x_1x_2 + 0.3x_2^2 + 0.2, 0.2x_1x_2 + 0.5x_2^2 + 0.3\}$. To see the justification for these Bellman equations, suppose for example that we select action a_1 for T_1. Then with probability 0.3 the process becomes extinct at this point, and with probability 0.7 there are two offspring of type T_1 and the process will become extinct iff both processes originating from the two offspring become extinct. Hence in the case of a_1 the extinction probability x_1 satisfies $x_1 = 0.7x_1^2 + 0.3$. The expressions for the other actions a_2, a_3 are derived similarly, and naturally we want to select the action that yields the maximum value among them. The intuitive reason why the true optimal extinction probabilities are given by the *least* fixed point of the equations was explained in the introduction. The LFP of the system in this example, and the vector of maximum extinction probabilities, is $q^* \approx (0.7, 0.486)$. The optimal strategy is to use action a_3 for T_1 and b_2 for T_2. □

A **1-exit Recursive Markov Decision Process** (1-RMDP) consists of a finite set of components $A_1, \ldots, A_k$, where each component A_i is essentially a finite-state MDP augmented with the ability to make recursive calls to itself and other components. Formally, each component A_i has a finite set N_i of nodes, which are partitioned into probabilistic nodes and controlled nodes, and a finite set B_i of "boxes" (or supernodes), where each box is mapped to some component. One node en_i is specified as the entry of the component A_i and one node ex_i as the exit of A_i.[3] The exit

[3]The restriction to having only one entry node is not important; any multi-entry RMDP can be efficiently trans-

node has no outgoing edges. All other nodes and the boxes have outgoing edges; the edges out of the probabilistic nodes and boxes are labeled with probabilities, where the sum of the probabilities out of the same node or box is equal to 1.

1-RMDP serve as a natural model for a class of recursive probabilistic programs. The components correspond to the recursive procedures, probabilistic nodes correspond to the probabilistic steps, controlled nodes correspond to the nonprobabilistic steps (e.g. branching steps), and the boxes correspond to the recursive calls. Execution of a 1-RMDP starts at some node, for example, the entry en_1 of component A_1. When the execution is at a probabilistic node v, then an edge out of v is chosen randomly according to the probabilities of the edges out of v. At a controlled node v, an edge out of v is chosen by a controller who seeks to optimize his objective. When the execution reaches a box b of A_i mapped to some component A_j, then the current component is suspended and a recursive call to A_j is initiated at its entry node en_j; if the call to A_j terminates, i.e. reaches eventually its exit node ex_j, then the execution of component A_i resumes from box b following an edge out of b chosen according to the probability distribution of the outgoing edges of b. Note that a call to a component can make further recursive calls, thus, at any point there is in general a stack of suspended recursive calls, and there can be an arbitrary number of such suspended calls; thus, a 1-RMDP induces generally an infinite-state MDP. The process terminates when the execution reaches the exit of the component of the initial node and there are no suspended recursive calls.

There are two types of 1-RMDPs with a termination objective: In a min 1-RMDP (resp. max 1-RMDP) the objective of the controller is to minimize (resp. maximize) the probability of termination. In principle, a controller can use the complete past history of the process and also use randomization (i.e. a mixed strategy) to select at each point when the execution reaches a controlled node which edge to select out of the node. As shown in [19] however, there is always an optimal strategy that is pure, stackless and memoryless, i.e., selects deterministically one edge out of each controlled node, the same one every time, independent of the stack of suspended calls and of the past history (including the starting node). From a given min or max 1-RMDP we can construct efficiently a minPPS or maxPPS, whose LFP yields the minimum or maximum termination probabilities for all the different possible starting vertices of the 1-RMDP [19]: There is one variable x_u for each node u, corresponding to the optimal termination probability starting at node u and two variables x_b, x'_b for each box b, corresponding to the optimal termination probabilities respectively when the recursive call for b is made (is initiated) and when it returns. The exit nodes ex_i have value $x_{ex_i} = 1$. The equation for each probabilistic node u whose outgoing edges (u, v) have probabilities p_{uv} is $x_u = \sum_v p_{uv} x_v$; the equation for the variable x'_b of a box b is $x'_b = \sum_v p_{bv} x_v$; the equation for the variable x_b of a box b mapped to component A_i with entry en_i is $x_b = x_{en_i} x'_b$; the equation for a controlled node u in a min 1-RMDP (resp. max 1-RMDP) is $x_u = \min\{x_v | (u, v) \in E\}$ (resp., $x_u = \max\{x_v | (u, v) \in E\}$. Conversely, from any given max/minPPS, we can efficiently construct a 1-RMDP whose optimal termination probabilities yield the LFP of the max/minPPS. The system of equations for a 1-RMDP has a particularly simple form. All max/minPPSs can be put in that form.

It is convenient to put max/minPPSs in the following simple form.

Definition 2.6. *A maxPPS in* **simple normal form (SNF)**, $x = P(x)$, *is a system of n equations in n variables $x_1, x_2, ...x_n$ where each $P_i(x)$ for $i = 1, 2, ...n$ is in one of three forms:*

- `Form L`*: $P(x)_i = a_{i,0} + \sum_{j=1}^{n} a_{i,j} x_j$, where $a_{i,j} \geq 0$ for all j, and such that $\sum_{j=0}^{n} a_{i,j} \leq 1$*

formed to an 1-entry RMDP. The restriction to 1-exit is very important: multi-exit RMDPs lead to undecidable termination problems, even for any non-trivial approximation of the optimal values [19].

- `Form Q`: $P(x)_i = x_j x_k$ *for some* j, k
- `Form M`: $P(x)_i = \max\{x_j, x_k\}$ *for some* j, k

We define **SNF form** *for minPPSs analogously: only the definition of "*`Form M`*" changes, replacing* max *with* min.

In the setting of a max/minPPS in SNF form, for simplicity in notation, when we talk about a policy, if $P_i(x)$ has form M, say $P_i(x) \equiv \max\{x_j, x_k\}$, then when it is clear from the context we will use $\sigma(i) = k$ to mean that the policy σ chooses x_k among the two choices x_j and x_k available in $P_i(x) \equiv \max\{x_j, x_k\}$.

Using similar techniques as in [18], it is easy to show that every max/minPPS can be transformed efficiently to one in SNF form; see the Appendix for the proof.

Proposition 2.7 (cf. Proposition 7.3 [18])**.** *Every max/minPPS, $x = P(x)$, can be transformed in P-time to an "*equivalent*" max/minPPS, $y = Q(y)$ in SNF form, such that $|Q| \in O(|P|)$. More precisely, the variables x are a subset of the variables y, the LFP of $x = P(x)$ is the projection of the LFP of $y = Q(y)$, and an optimal policy (respectively, ϵ-optimal policy) for $x = P(x)$ can be obtained in P-time from an optimal (resp., ϵ-optimal) policy of $y = Q(y)$.*

Thus from now on, and for the rest of this paper *we assume, without loss of generality, that all max/minPPSs are in SNF normal form.*

The *dependency graph* of a max/minPPS $x = P(x)$ is a directed graph G that has one node for every variable, and has an edge $x_i \to x_j$ iff the variable x_j appears in $P_i(x)$. We say that the system $x = P(x)$ is strongly connected if its dependency graph is strongly connected, i.e., if every node has a directed path to every other node.

We now summarize some of the main previous results on PPSs and max/minPPSs.

Proposition 2.8 (see [19]; and see the Appendix of this paper for more efficient P-time algorithms)**.** *There is a P-time algorithm that, given a minPPS or maxPPS, $x = P(x)$, over n variables, with LFP $q^* \in \mathbb{R}^n_{\geq 0}$, determines for every $i = 1, \ldots, n$ whether $q^*_i = 0$ or $q^*_i = 1$ or $0 < q^*_i < 1$.*

Thus, given a max/minPPS we can find in P-time all the variables x_i such that $q^*_i = 0$ or $q^*_i = 1$, remove them and their corresponding equations $x_i = P_i(x)$, and substitute their values on the RHS of the remaining equations. This yields a new max/minPPS, $x' = P'(x')$, where its LFP solution, q'^*, is $0 < q'^* < 1$, which corresponds to the remaining coordinates of q^*. Thus, it suffices to focus our attention to systems whose LFP is strictly between 0 and 1.

The decision problem of determining whether a coordinate q^*_i of the LFP is $\geq 1/2$ (or whether $q^*_i \geq r$ for any other given bound $r \in (0, 1)$) is at least as hard as the `sqrt-sum` and the `PosSLP` problems even for PPS (without the min and max operator) [18] and hence it is highly unlikely that it can be solved in P.

The problem of approximating efficiently the LFP of a PPS was solved recently in [12, 13], by using Newton's method after elimination of the variables with value 0 and 1.

Definition 2.9. *For a PPS $x = P(x)$ we use $P'(x)$ to denote the Jacobian matrix of partial derivatives of $P(x)$, i.e., $P'(x)_{i,j} := \frac{\partial P_i(x)}{\partial x_j}$. For a point $x \in \mathbb{R}^n$, if $(I - P'(x))$ is non-singular, then we define one Newton iteration at x via the operator:*

$$\mathcal{N}(x) = x + (I - P'(x))^{-1}(P(x) - x)$$

Given a max/minPPS, x=P(x), and a policy σ, we use $\mathcal{N}_\sigma(x)$ to denote the Newton operator of the PPS $x = P_\sigma(x)$; i.e., if $(I - P'_\sigma(x))$ is non-singular at a point $x \in \mathbb{R}^n$, then $\mathcal{N}_\sigma(x) = x + (I - P'_\sigma(x))^{-1}(P_\sigma(x) - x)$.

Theorem 2.10 (Theorem 3.2 and Corollary 4.5 of [13]). *Let $x = P(x)$ be a PPS with rational coefficients in SNF form which has least fixed point $0 < q^* < 1$. If we conduct iterations of Newton's method as follows: $x^{(0)} := 0$, and for $k \geq 0$: $x^{(k+1)} := \mathcal{N}(x^{(k)})$, then the Newton operator $\mathcal{N}(x^{(k)})$ is defined for all $k \geq 0$, and for any $j > 0$:*

$$\|q^* - x^{(j+4|P|)}\|_\infty \leq 2^{-j}$$

where $|P|$ is the total bit encoding length of the system $x = P(x)$.

Furthermore, there is an algorithm (based on suitable rounding of Newton iterations) which, given a PPS, $x = P(x)$, and given a positive integer j, computes a rational vector $v \in [0,1]^n$, such that $||q^* - v||_\infty \leq 2^{-j}$, and which runs in time polynomial in $|P|$ and j in the standard Turing model of computation.

The proof of the theorem involves a number of technical lemmas on PPSs and Newton's method, several of which we will also need in this paper, some of them in strengthened form, and which we include in the appendix. The following lemma summarizes key properties of the Newton operator for PPS that are crucial for the correctness and the polynomial running time.

Lemma 2.11 (Combining Lemmas 3.9, 3.5 and Theorem 3.7 of [13]). *Let $x = P(x)$ be a PPS in SNF with $0 < q^* < 1$. For any $0 \leq x \leq q^*$ and $\lambda > 0$, the operator $\mathcal{N}(x)$ is defined (i.e. the matrix $I - P'(x)$ is non-singular), $\mathcal{N}(x) \leq q^*$, and if $q^* - x \leq \lambda(1 - q^*)$ then $q^* - \mathcal{N}(x) \leq \frac{\lambda}{2}(1 - q^*)$.*

If we knew an optimal policy τ for a max/minPPS, $x = P(x)$, then we would be able to solve the problem of computing the LFP for a max/minPPS using the algorithm in [13] for approximating q^*_τ, because we know $q^*_\tau = q^*$. Unfortunately, we do not know which policy is optimal. There are exponentially many policies, so it would be inefficient to run this algorithm using every policy. (And even if we did do so for each possible policy, we would only be able to ϵ-approximate the values q^*_σ for each policy σ using the results of [13], for say, $\epsilon = 2^{-j}$ for some chosen j, and therefore we could only be sure that a particular policy that yields the best result is, say, (2ϵ)-optimal, but it may not not necessarily be optimal.) In fact, as we will see, it is probably impossible to identify an optimal policy in polynomial time.

Our goal instead will be to find an iteration $I(x)$ for max/minPPS, that has similar properties to the Newton operator for PPS, i.e., that can be computed efficiently for a given x and for which we can prove a similar property to Lemma 2.11, i.e. such that if $q^* - x \leq \lambda(1 - q^*)$, then $q^* - I(x) \leq \frac{\lambda}{2}(1 - q^*)$. Once we do so, we will be able to adapt and extend results from [13] to get a polynomial time algorithm for the problem of approximating the LFP q^* of a max/minPPS.

3 Generalizing Newton's method using linear programming

If a max/minPPS, $x = P(x)$, has no equations of form Q, then it amounts to precisely the Bellman equations for an ordinary finite-state Markov Decision Process with the objective of maximizing/minimizing reachability probabilities. It is well known that we can compute the exact (rational) optimal values for such finite-state MDPs, and thus the exact LFP, q^*, for such a max(min)-linear

systems, using linear programming (see, e.g., [29, 7]). Computing the LFP of max/minPPSs is clearly a generalization of this finite-state MDP problem to the infinite-state setting of branching and recursive MDPs.

If we have no equations of form M, we have a PPS, which we can solve in P-time, as shown recently in [13]. The algorithm first preprocesses the system to identify the variables that have value 0 or 1 in the LFP, removes them from the system, and then applies Newton's method on the remaining system. Recall that an iteration of Newton's method works by approximating the system of equations by a linear system, which is a linearization of the system around the current point, and solving this linear system to obtain the new point.

For maxPPS(or minPPS) we employ a similar approach. We first identify the variables that have value 0 or 1 in the LFP using the algorithms of [19] or the improved algorithms in the Appendix. We remove these variables, substituting their value, and thereby obtain a reduced system on the remaining variables whose LFP q^* satisfies $0 < q^* < 1$. We compute (approximately) the LFP of the remaining maxPPS (or minPPS) by an iterative algorithm, which we call *Generalized Newton's Method* (GNM). For this purpose, we define an analogous "approximate" system of equations at the current point, which has both linear equations and equations involving the max (or min) function. We show that we can solve the equations that arise from each iteration of GNM using linear programming. We then show that a polynomial (in fact, linear) number of iterations are enough to approximate the desired LFP solution, and that it suffices to carry out the computations with polynomial precision.

We begin by defining formally the max/min linear equations that should be solved by one iteration of "Generalized Newton's Method" (GNM), applied at a point y. Recall that we assume w.l.o.g. throughout that max/minPPSs and PPSs are in SNF.

Definition 3.1. *For a max/minPPS, $x = P(x)$, with n variables, the* **linearization of $P(x)$ at a point $\mathbf{y} \in \mathbb{R}^n$,** *is a system of max/min linear functions denoted by $P^y(x)$, which has the following form:*

if $P_i(x)$ has form L or M, then $P_i^y(x) = P_i(x)$, and
if $P_i(x)$ has form Q, i.e., $P_i(x) = x_j x_k$ for some j,k, then

$$P_i^y(x) = y_j x_k + x_j y_k - y_j y_k$$

Example 3.1. Consider the following minPPS $x = P(x)$:

$$x_1 = 0.2x_2 + 0.3x_3 + 0.5; \quad x_2 = 0.4x_1 + 0.1x_3 + 0.5x_4; \quad x_3 = \min(x_2, x_5); \quad x_4 = x_1x_3; \quad x_5 = x_1^2$$

Its linearization $x = P^y(x)$ at the point $y = (0.8, 0.3, 0.4, 0.2, 0.5)$ is

$$\begin{aligned} x_1 &= 0.2x_2 + 0.3x_3 + 0.5; \quad x_2 = 0.4x_1 + 0.1x_3 + 0.5x_4; \quad x_3 = \min(x_2, x_5); \\ x_4 &= 0.4x_1 + 0.8x_3 - 0.32; \; x_5 = 1.6x_1 - 0.64. \qquad \square \end{aligned}$$

We define distinct iteration operators for a maxPPS and a minPPS, both of which we shall refer to with the overloaded notation $I(y)$. These operators will serve as the basis for a *Generalized Newton's Method* to be applied to maxPPSs and minPPSs, respectively.

Definition 3.2. *For a maxPPS, $x = P(x)$, with LFP q^*, such that $0 < q^* < 1$, and for a real vector y such that $0 \leq y \leq q^*$, we define the operator $I(y)$ to be the* unique *optimal solution, $a \in \mathbb{R}^n$, to the following mathematical program[4]:* Minimize: $\sum_i a_i$; Subject to: $P^y(a) \leq a$.

[4]Note that we do not constrain the variables a to be non-negative in the mathematical programs corresponding to the operator I(y) for both maxPPSs and minPPSs.

For a minPPS, $x = P(x)$, with LFP q^*, such that $0 < q^* < 1$, and for a real vector y such that $0 \leq y \leq q^*$, we define the operator $I(y)$ to be the unique *optimal solution $a \in \mathbb{R}^n$ to the following mathematical program:* Maximize: $\sum_i a_i$; Subject to: $P^y(a) \geq a$.

A priori, it is not obvious that the above definitions of $I(y)$ for maxPPSs and minPPSs are well-defined, i.e., that the mathematical programs are feasible and have a unique optimal solution. We will show in the following subsections that this is indeed the case. We will also show that the mathematical programs can be expressed as linear programs, and thus can be solved in polynomial time.

Example 3.2. For the minPPS $x = P(x)$ of the previous example, and the vector $y = (0.8, 0.3, 0.4, 0.2, 0.5)$, the corresponding mathematical program is:
Maximize: $\sum_{i=1}^{5} a_i$; Subject to:

$$\begin{aligned} a_1 &\leq 0.2a_2 + 0.3a_3 + 0.5; \quad a_2 \leq 0.4a_1 + 0.1a_3 + 0.5a_4; \quad a_3 \leq \min(a_2, a_5); \\ a_4 &\leq 0.4a_1 + 0.8a_3 - 0.32; \; a_5 \leq 1.6a_1 - 0.64 \end{aligned}$$

The third constraint can be written equivalently as the conjunction of the inequalities $a_3 \leq a_2$ and $a_3 \leq a_5$, which yields a Linear Program. The LP has a unique optimal solution $(0.85, 0.7, 0.7, 0.58, 0.72)$, and this vector is $I(y)$. Note that this vector satisfies $a = P^y(a)$. □

Now we can give a polynomial time algorithm, in the Turing model of computation, for approximating the LFP for a max/minPPS, to within any desired precision. First, compute the set of variables that have value 0 or 1 in the LFP using the algorithms of [19], or the improved algorithms of the Appendix; remove these variables from the system, yielding a remaining system whose LFP q^* satisfies $0 < q^* < 1$. Then apply the following algorithm to compute iteratively a sequence of points $x^{(k)}, k = 0, 1, 2 \ldots$, starting from $x^{(0)} := 0$, rounding down every point along the computation to h bits of precision. The number of iterations and the rounding parameter h depend on the desired number of bits of precision in the approximation of the LFP.

Algorithm (Generalized Newton's Method with rounding)
Start with $x^{(0)} := 0$;
For each $k \geq 0$ compute $x^{(k+1)}$ from $x^{(k)}$ as follows:

1. Calculate $I(x^{(k)})$ by solving the following LP:
   Minimize: $\sum_i x_i$; *Subject to:* $P^{x^{(k)}}(x) \leq x$, if $x = P(x)$ is a maxPPS,
   or:
   Maximize: $\sum_i x_i$; *Subject to:* $P^{x^{(k)}}(x) \geq x$, if $x = P(x)$ is a minPPS.

2. For each coordinate $i = 1, 2, ...n$, set $x_i^{(k+1)}$ to be the maximum (non-negative) multiple of 2^{-h} which is $\leq \max\{0, I(x^{(k)})_i\}$. (In other words, we round $I(x^{(k)})$ down to the nearest 2^{-h} and ensure it is non-negative.)

Example 3.3. Consider the minPPS of the previous Examples. Applying the algorithm of [19] or of the Appendix yields that all variables have value strictly between 0 and 1 in the LFP, thus no variable is eliminated. We start the Generalized Newton's method with $x^{(0)} := 0$. The LP is the same as that of Example 3.2 except for the last two constraints (corresponding to the form-Q equations of the minPPS), which are $a_4 \leq 0$ and $a_5 \leq 0$. Solving the LP yields the next point $x^{(1)} \approx (0.54, 0.22, 0, 0, 0)$. The LP in the next iteration changes again only in the last two constraints and yields the next point $x^{(2)} \approx (0.73, 0.47, 0.47, 0.25, 0.50)$. After a few more iterations

we get $x^{(5)} \approx (0.897, 0.795, 0.795, 0.713, 0.805)$ and $x^{(6)} \approx (0.8999, 0.7999, 0.7999, 0.7198, 0.8099)$. The actual LFP is $q^* = (0.9, 0.8, 0.8, 0.72, 0.81)$. In this example, Value Iteration takes about 200 iterations to reach the accuracy of $x^{(5)}$ and 400 iterations for $x^{(6)}$ (of course the iterations themselves are simpler). □

We shall prove the following theorem:

Theorem 3.3. *Given any max/minPPS, $x = P(x)$, with LFP $0 < q^* < 1$, if we use the above algorithm with rounding parameter $h = j + 2 + 4|P|$, then the iterations are all defined, and for every $k \geq 0$ we have $0 \leq x^{(k)} \leq q^*$, and furthermore after $h - 1 = j + 1 + 4|P|$ iterations we have:*

$$\|q^* - x^{(j+1+4|P|)}\|_\infty \leq 2^{-j}$$

.

Corollary 3.4. *Given any max/minPPS, $x = P(x)$, with LFP q^*, and given any integer $j > 0$, there is an algorithm that computes a rational vector $v \leq q^*$ with $\|q^* - v\|_\infty \leq 2^{-j}$, in time polynomial in $|P|$ and j.*

The rest of this Section is devoted to proving Theorem 3.3 and the corollary. The Section is organized as follows. In Section 3.1 we give some basic properties of the linearization of a max/minPPS (their proofs are given in the Appendix). In 3.2 we state the key properties of the operator $I(\cdot)$ for an iteration of the Generalized Newton's method. In Section 3.3 we analyze the operator for maxPPS and in Section 3.4 for minPPS and prove its key properties. Finally in Section 3.5 we put everything together and prove Theorem 3.3 and Corollary 3.4, showing that the algorithm approximates the LFP within any desired precision in polynomial time in the Turing model.

3.1 Linearizations of max/minPPSs and their properties

Let $x = P(x)$ be a maxPPS or minPPS. For any policy σ, we can consider the linearization of the corresponding PPS, $x = P_\sigma(x)$.

Definition 3.5. $P^y_\sigma(x) := (P_\sigma)^y(x)$.

Note that the linearization $P^y(x)$ only changes equations of form Q, and using a policy σ only changes equations of form M, so these operations are independent in terms of the effects they have on the underlying equations, and thus $P^y_\sigma(x) \equiv (P_\sigma)^y(x) = (P^y)_\sigma(x)$.

We first state some basic properties of linearizations of PPS (without max or min); see the Appendix for the proofs.

Lemma 3.6. *Let $x = P(x)$ be any PPS. For any $y \in \mathbb{R}^n$, let $(P^y)'(x)$ denote the Jacobian matrix of $P^y(x)$. Then for any $x \in \mathbb{R}^n$, we have $(P^y)'(x) = P'(y)$.*

Lemma 3.7. *If $x = P(x)$ is any PPS, then for any $x, y \in \mathbb{R}^n$, $P^y(x) = P(y) + P'(y)(x - y)$.*

An iteration of Newton's method on $x = P_\sigma(x)$ at a point y solves a system of linear equations that can be expressed in terms of $P^y_\sigma(x)$. The next lemma establishes this basic fact in part *(i)*. In part *(ii)* it provides us with conditions under which we are guaranteed to be doing "at least as well" as one such Newton iteration. (See the Appendix for the proof.)

Lemma 3.8. *Let $x = P(x)$ be any max/minPPS. Suppose that the matrix inverse $(I - P'_\sigma(y))^{-1}$ exists and is non-negative, for some policy σ, and some $y \in \mathbb{R}^n$. Then*

(i) $\mathcal{N}_\sigma(y)$ *is defined, and is equal to the unique point* $a \in \mathbb{R}^n$ *such that* $P^y_\sigma(a) = a$.

(ii) For any vector $x \in \mathbb{R}^n$*:*
If $P^y_\sigma(x) \geq x$*, then* $x \leq \mathcal{N}_\sigma(y)$.
If $P^y_\sigma(x) \leq x$*, then* $x \geq \mathcal{N}_\sigma(y)$.

3.2 Key properties of the iteration operator of GNM

Definition 3.2 defines the iteration operator $I(y)$ as the unique optimal solution of a mathematical program. We first observe that this mathematical program can be expressed as a linear program, for both maxPPSs and minPPSs.

Proposition 3.9. *Given a max/minPPS,* $x = P(x)$*, with LFP* q^**, and given a rational vector* y*,* $0 \leq y \leq q^*$*, the constrained optimization problem (i.e., mathematical program) defining* $I(y)$ *can be described by a LP whose encoding size is polynomial (in fact, linear) in both* $|P|$ *and the encoding size of the rational vector* y*. Thus, we can compute the (unique) optimal solution* $I(y)$ *to such an LP (assuming it exists, and is unique) in P-time.*

Proof. For a maxPPS (minPPS), the definition of $I(y)$ asks us to maximize (minimize) a linear objective, $\sum_i a_i$, subject to the constraints $P^y(a) \leq a$ ($P^y(a) \geq a$, respectively). All of these constraints are linear, except the constraints of form M. For a maxPPS, if $(P^y(a))_i$ is of form M, then the corresponding constraint is an inequality of the form $\max\{a_j, a_k\} \leq a_i$. Such an inequality is equivalent to, and can be replaced by, the two linear inequalities: $a_j \leq a_i$ and $a_k \leq a_i$. Likewise, for a minPPS, if $(P^y(a))_i$ is of form M, then the corresponding constraint is an inequality of the form $\min\{a_j, a_k\} \geq a_i$. Again, such an inequality is equivalent to, and can be replaced by, two linear inequalities: $a_j \geq a_i$ and $a_k \geq a_i$.

Thus, for a rational vector y whose encoding length is $\texttt{size}(y)$, the operator $I(y)$ can be formulated (for both maxPPSs and minPPSs) as a problem of computing the unique optimal solution to a linear program whose encoding size is polynomial (in fact, linear) in $|P|$ and in $\texttt{size}(y)$. □

The following proposition lists key properties of the operator $I(y)$. In particular (part 1), the operator is well-defined if $0 < y < q^*$, i.e. the mathematical program is feasible and has a unique optimal solution. Furthermore (part 2), the iteration makes in some sense good progress towards the LFP q^*; this part is useful in establishing the speed of convergence of GNM.

Proposition 3.10. *Let* $x = P(x)$ *be a max/minPPS, with LFP* q^**, such that* $0 < q^* < 1$*. For any* $0 \leq y \leq q^*$*:*

1. $I(y)$ *is well-defined, and* $I(y) \leq q^*$*, and:*
2. *For any* $\lambda > 0$*, if* $q^* - y \leq \lambda(1 - q^*)$ *then* $q^* - I(y) \leq \frac{\lambda}{2}(1 - q^*)$.

We shall prove the proposition separately for maxPPSs and minPPSs in the following two subsections.

3.3 An iteration of Generalized Newton's Method (GNM) for maxPPSs

In this subsection we will analyze the operator $I(y)$ for a maxPPS and prove its key properties given in Proposition 3.10. For a maxPPS, $x = P(x)$, we know by Theorem 2.5 that there exists an optimal policy, τ, such that $q^* = q^*_\tau \geq q^*_\sigma$ for any policy σ. The next lemma implies part 1 of Proposition 3.10 for maxPPS:

Lemma 3.11. *If $x = P(x)$ is a maxPPS, with LFP solution $0 < q^* < 1$, and y is a real vector with $0 \leq y \leq q^*$, then $x = P^y(x)$ has a least fixed point solution, denoted μP^y, with $\mu P^y \leq q^*$. Furthermore, the operator $I(y)$ is well-defined, $I(y) = \mu P^y \leq q^*$, and for any optimal policy τ, $I(y) = \mu P^y \geq \mathcal{N}_\tau(y)$.*

Proof. Recall that (by Proposition 3.9) the mathematical program that "defines" $I(y)$ can be written equivalently as an LP:

$$\textit{Minimize: } \sum_i a_i \ ; \qquad \textit{Subject to: } \quad P^y(a) \leq a \tag{1}$$

Firstly, we show that the LP constraints $P^y(a) \leq a$ in the definition of $I(y)$ are *feasible*. We do so by showing that actually $P^y(q^*) \leq q^*$. At any coordinate i, if $P_i(x)$ has form M or L, then $P_i^y(q^*) = P_i(q^*) = q_i^*$. Otherwise, $P_i(x)$ has form Q, i.e., $P_i(x) = x_j x_k$, and then

$$\begin{aligned} P_i^y(q^*) &= q_j^* y_k + y_j q_k^* - y_j y_k \\ &= q_j^* q_k^* - (q_j^* - y_j)(q_k^* - y_k) \\ &\leq q_i^* \qquad\qquad (\text{since } y \leq q^*) \end{aligned}$$

Next we show that the LP (1) defining $I(y)$ is *bounded*. Recall that, by Theorem 2.5, there is always an optimal policy for any maxPPS, $x = P(x)$.

Claim 3.12. *Let $x = P(x)$ be any maxPPS, with $0 < q^* < 1$, and let τ be any optimal policy for $x = P(x)$. For any y such that $0 \leq y \leq q^*$, we have that $\mathcal{N}_\tau(y)$ is defined, and for any vector a, if $P^y(a) \leq a$ then $\mathcal{N}_\tau(y) \leq a$. In particular, $\mathcal{N}_\tau(y) \leq q^*$.*

Proof. Recall, from our definition of an optimal policy, that $q^* = q_\tau^*$ is also the least non-negative solution to $x = P_\tau(x)$. So we can apply Lemma B.3 (in the Appendix) using $x = P_\tau(x)$ and $y \leq q^*$ to deduce that $(I - P'_\tau(y))^{-1}$ exists and is non-negative. Thus $\mathcal{N}_\tau(y)$ is defined. Now, by applying Lemma 3.8 (ii), to show that $a \geq \mathcal{N}_\tau(y)$ all we need to show is that $P_\tau^y(a) \leq a$. But recalling that $x = P(x)$ is a maxPPS, by the definition of $P^y(x)$ and $P_\tau^y(x)$, we have that $P_\tau^y(a) \leq P^y(a) \leq a$. We have just shown before this Claim that $P^y(q^*) \leq q^*$, and thus $\mathcal{N}_\tau(y) \leq q^*$. □

Thus the LP (1) defining $I(y)$ is both feasible and bounded, hence it has an optimal solution. To show that $I(y)$ is well-defined, all that remains is to show that this optimal solution is unique. In the process, we will also show that $I(y)$ defines precisely the *least fixed point* solution of $x = P^y(x)$, which we denote by μP^y.

Firstly, we claim that for any optimal solution b to the LP (1), it must be the case that $P^y(b) = b$. Suppose not. Then there exists i such that $P^y(b)_i < b_i$, then we can define a new vector b', such that $b'_i = P^y(b)_i$ and $b'_j = b_j$ for all $j \neq i$. By monotonicity of $P^y(x)$, it is clear that $P^y(b') \leq b'$, and thus that b' is a feasible solution to the LP (1). But $\sum_i b'_i < \sum_i b_i$, contradicting the assumption that b is an optimal solution to the LP (1).

Secondly, we claim that there is a unique optimal solution. Suppose not: suppose b and c are two distinct optimal solution to the LP (1). Define a new vector d by $d_i = \min\{b_i, c_i\}$, for all i. Clearly, $d \leq b$ and $d \leq c$. Thus by the monotonicity of $P^y(x)$, for all i $P^y(d)_i \leq P^y(b)_i = b_i$, and likewise $P^y(d)_i \leq P^y(c)_i = c_i$. Thus $P^y(d) \leq d$, and d is a feasible solution to the LP. But since b and c are distinct, and yet $\sum_i b_i = \sum_i c_i$, we have that $\sum_i d_i < \sum_i b_i = \sum_i c_i$, contradicting the optimality of both b and c.

We have thus established that $I(y)$ defines the unique *least fixed point* solution of $x = P^y(x)$, which we denote also by μP^y. Since q^* is also a solution of the LP, we have $\mu P^y \leq q^*$.

Finally, by Claim 3.12, it must be the case that $I(y) = \mu P^y \geq \mathcal{N}_\tau(y)$, where τ is any optimal policy for $x = P(x)$. □

We next establish part 2 of Proposition 3.10 for maxPPS.

Lemma 3.13. *Let $x = P(x)$ be a maxPPS with $0 < q^* < 1$. For any $0 \leq y \leq q^*$ and $\lambda > 0$, we have $I(y) \leq q^*$, and furthermore if:*

$$q^* - y \leq \lambda(1 - q^*)$$

then

$$q^* - I(y) \leq \frac{\lambda}{2}(1 - q^*)$$

Proof. Let τ be an optimal policy (which exists by Theorem 2.5). The least fixed point solution of the PPS $x = P_\tau(x)$ is q^*. From our assumptions, Lemma 2.11 gives that $q^* - \mathcal{N}_\tau(y) \leq \frac{\lambda}{2}(1 - q^*)$. But by Lemma 3.11 $\mathcal{N}_\tau(y) \leq I(y) \leq q^*$. The claim follows. □

Proposition 3.10 for maxPPSs follows from Lemmas 3.11 and 3.13.

3.4 An iteration of GNM for minPPSs

In this subsection we will prove the key properties of the operator $I(y)$ for minPPS (Proposition 3.10). Our proof of the minPPS version will be somewhat different, because it turns out we can not use the same argument as for maxPPS, based on LPs, to prove that $I(y)$ is well-defined. Fortunately, in the case of minPPSs, we can show that $(I - P_\sigma(y))^{-1}$ exists and is non-negative for *any* policies σ, at those points y that are of interest. And we can use this to show that there is *some* policy, σ, such that $I(y)$ is equivalent to an iteration of Newton's method at y after fixing the policy σ. We shall establish the existence of such a policy using a policy improvement argument, instead of just using the LP, as we did for maxPPSs. (Note that the policy improvement algorithm may not be an efficient (P-time) way to compute it, and we do not claim it is. We only use policy improvement as an argument in the proof of existence of a suitable policy σ.)

Lemma 3.14. *For a minPPS, $x = P(x)$, with LFP $0 < q^* < 1$, for any $0 \leq y \leq q^*$ and any policy σ, $(I - P_\sigma(y))^{-1}$ exists and is non-negative. Thus $\mathcal{N}_\sigma(y)$ is defined.*

Proof. We show first that the LFP of $x = P_\sigma(x)$, denoted q^*_σ, satisfies $q^* \leq q^*_\sigma$. To see this, note that by Theorem 2.5, there is an optimal policy τ with $q^*_\tau = q^*$. But we defined an optimal policy for a minPPS as one with $q^*_\tau \leq q^*_\upsilon$ for any policies υ. Therefore $q^* = q^*_\tau \leq q^*_\sigma$.

Since $0 < q^* < 1$, and $0 \leq y \leq q^* \leq q^*_\sigma$, we have $q^*_\sigma > 0$, $0 \leq y \leq q^*_\sigma$, and $y < 1$. It follows from Lemma B.3 of the Appendix, applied to the PPS $x = P_\sigma(x)$, that $(I - P_\sigma(y))^{-1}$ exists and is non-negative, and hence $\mathcal{N}_\sigma(y)$ is defined. □

Lemma 3.15. *Given a minPPS, $x = P(x)$, with LFP $0 < q^* < 1$, and a vector y with $0 \leq y \leq q^*$, there is a policy σ such that $P^y(\mathcal{N}_\sigma(y)) = \mathcal{N}_\sigma(y)$.*

Proof. We use a policy (strategy) improvement "algorithm" to prove this. Start with any policy σ_1. At step i, suppose we have a policy σ_i.

For notational simplicity, in the following we use the abbreviation: $z = \mathcal{N}_{\sigma_i}(y)$. By Lemma 3.8, $P^y_{\sigma_i}(z) = z$. So we have $P^y(z) \leq z$. If $P^y(z) = z$, then *stop*: we are done.

Otherwise, to construct the next strategy σ_{i+1}, take the smallest j such that $(P^y(z))_j < z_j$. Note that $P_j(x)$ has form M, because otherwise $(P(x))_j = (P_{\sigma_i}(x))_j$. Thus, there is some variable x_k with $P_j(x) = \min\{x_k, x_{\sigma_i(j)}\}$ and $z_k < z_{\sigma_i(j)}$. Define σ_{i+1} to be:

$$\sigma_{i+1}(l) = \begin{cases} \sigma_i(l) & \text{if } l \neq j \\ k & \text{if } l = j \end{cases}$$

Then $(P^y_{\sigma_{i+1}}(z))_j < z_j$, but for every other coordinate $l \neq j$, $(P^y_{\sigma_{i+1}}(z))_l = (P^y_{\sigma_i}(z))_l = z_l$. Thus

$$P^y_{\sigma_{i+1}}(z) \leq z \tag{2}$$

By Lemma 3.14, $\mathcal{N}_{\sigma_{i+1}}(y)$ is defined. Moreover, the inequality (2), together with Lemma 3.8 (ii), yields that $\mathcal{N}_{\sigma_{i+1}}(y) \leq z$. But $\mathcal{N}_{\sigma_{i+1}}(y) \neq z$ because $P^y_{\sigma_{i+1}}(z) \neq z$ whereas, by Lemma 3.8 (i), we have $P^y_{\sigma_{i+1}}(\mathcal{N}_{\sigma_{i+1}}(y)) = \mathcal{N}_{\sigma_{i+1}}(y)$.

Thus this algorithm gives us a sequence of policies $\sigma_1, \sigma_2 ...$ with $\mathcal{N}_{\sigma_1}(y) \geq \mathcal{N}_{\sigma_2}(y) \geq \mathcal{N}_{\sigma_3}(y) \geq ...$, where furthermore each step must strictly decrease at least one coordinate of $\mathcal{N}_{\sigma_i}(y)$. It follows that $\sigma_i \neq \sigma_j$, unless $i = j$. There are only finitely many policies. So the sequence must be finite, and the algorithm terminates. But it only terminates when we reach a σ_i with $P^y(\mathcal{N}_{\sigma_i}(y)) = \mathcal{N}_{\sigma_i}(y)$. □

We note that the analogous policy improvement algorithm might fail to work for maxPPSs, as we might reach a policy σ_i where $(I - P_{\sigma_i}(x))^{-1}$ does not exist, or has a negative entry.

The next Lemma shows that this policy improvement algorithm always produces a coordinate-wise minimal Newton iterate over all policies.

Lemma 3.16. *For a minPPS, $x = P(x)$, with LFP $0 < q^* < 1$, if $0 \leq y \leq q^*$ and σ is a policy such that $P^y(\mathcal{N}_\sigma(y)) = \mathcal{N}_\sigma(y)$, then:*

(i) For any policy σ', $\mathcal{N}_{\sigma'}(y) \geq \mathcal{N}_\sigma(y)$.

(ii) For any $x \in \mathbb{R}^n$ with $P^y(x) \geq x$, we have $x \leq \mathcal{N}_\sigma(y)$.

(iii) For any $x \in \mathbb{R}^n$ with $P^y(x) \leq x$, we have $x \geq \mathcal{N}_\sigma(y)$.

(iv) $\mathcal{N}_\sigma(y)$ is the unique fixed point of $x = P^y(x)$.

(v) $\mathcal{N}_\sigma(y) \leq q^*$.

Proof. Note firstly that by Lemma 3.14, for any policy σ, $(I - P'_\sigma(y))^{-1}$ exists and is non-negative, and $\mathcal{N}_\sigma(y)$ is defined.

(i) Consider $P^y_{\sigma'}(\mathcal{N}_\sigma(y))$. Note that $P^y_{\sigma'}(\mathcal{N}_\sigma(y)) \geq P^y(\mathcal{N}_\sigma(y)) = \mathcal{N}_\sigma(y)$ by assumption. Thus, by Lemma 3.8 (ii), $\mathcal{N}_\sigma(y) \leq \mathcal{N}_{\sigma'}(y)$.

(ii) $P^y_\sigma(x) \geq P^y(x) \geq x$, so by Lemma 3.8 (ii), $x \leq \mathcal{N}_\sigma(y)$.

(iii) If $P^y(x) \leq x$, then there a policy σ' with $P^y_{\sigma'}(x) \leq x$, and by Lemma 3.8 (ii), $x \geq \mathcal{N}_{\sigma'}(y)$. So using part (i) of this Lemma, $x \geq \mathcal{N}_{\sigma'}(y) \geq \mathcal{N}_\sigma(y)$.

(iv) By assumption, $\mathcal{N}_\sigma(y)$ is a fixed point of $x = P^y(x)$. We just need uniqueness. If $P^y(q) = q$, then by parts (ii) and (iii) of this Lemma, $q \leq \mathcal{N}_\sigma(y)$ and $q \geq \mathcal{N}_\sigma(y)$, i.e., $q = \mathcal{N}_\sigma(y)$.

(v) Consider an optimal policy τ, for the minPPS, $x = P(x)$. From Lemma 2.11, if follows that $\mathcal{N}_\tau(y) \leq q^*_\tau = q^*$. And then part (i) of this Lemma, gives us that $\mathcal{N}_\sigma(y) \leq \mathcal{N}_\tau(y) \leq q^*$.

□

We can show now part 1 of Proposition 3.10. Recall the LP that "defines" $I(y)$, for a minPPS:

$$\textit{Maximize:} \quad \sum_i a_i \ ; \qquad \textit{Subject to:} \quad P^y(a) \geq a \tag{3}$$

Lemma 3.17. *For a minPPS, $x = P(x)$, with LFP $0 < q^* < 1$, and for $0 \leq y \leq q^*$, there is a unique optimal solution, which we call $I(y)$, to the LP (3), and furthermore $I(y) = \mathcal{N}_\sigma(y)$ for some policy σ, $P^y(I(y)) = I(y)$, and $I(y) \leq q^*$.*

Proof. By Lemma 3.15, there is a σ such that $P^y(\mathcal{N}_\sigma(y)) = \mathcal{N}_\sigma(y)$. So $\mathcal{N}_\sigma(y)$ is a feasible solution of $P^y(a) \geq a$. Let a by any solution of $P^y(a) \geq a$. By Lemma 3.16 (ii), $a \leq \mathcal{N}_\sigma(y)$. Consequently $\sum_{i=1}^n a_i \leq \sum_{i=1}^n (\mathcal{N}_\sigma(y))_i$ with equality only if $a = \mathcal{N}_\sigma(y)$. So $\mathcal{N}_\sigma(y)$ is the unique optimal solution of the LP (3) and $I(y) = \mathcal{N}_\sigma(y)$. By Lemma 3.16 (iv), $I(y) \leq q^*$. □

In the maxPPS case, we had an iteration that was at least as good as iterating with the optimal policy. Here we have an iteration that is at least as bad! Nevertheless, we shall see that it is good enough. In the maxPPS case, the analog of Lemma 2.11, Lemma 3.13, thus followed from Lemma 2.11. Here we crucially need the following stronger result for PPS than Lemma 2.11; its proof is given in the Appendix.

Lemma 3.18. *If $x = P(x)$ is a PPS and we are given $x, y \in \mathbb{R}^n$ with $0 \leq x \leq y \leq P(y) \leq 1$, and if the following conditions hold:*

$$\lambda > 0 \quad \textit{and} \quad y - x \leq \lambda(1 - y) \quad \textit{and} \quad (I - P'(x))^{-1} \textit{ exists and is non-negative,} \tag{4}$$

then $y - \mathcal{N}(x) \leq \frac{\lambda}{2}(1 - y)$.

(Note that we cannot conclude that $y - \mathcal{N}(x) \geq 0$.)

We can show now part 2 of Proposition 3.10.

Lemma 3.19. *Let $x = P(x)$ be a minPPS, with LFP $0 < q^* < 1$. For any $0 \leq x \leq q^*$ and $\lambda > 0$, if:*

$$q^* - x \leq \lambda(1 - q^*)$$

then

$$q^* - I(x) \leq \frac{\lambda}{2}(1 - q^*)$$

Proof. By Lemma 3.17, there is a policy σ with $I(x) = \mathcal{N}_\sigma(x)$. We then apply Lemma 3.18 to $x = P_\sigma(x)$, x, and q^* instead of y. Observe that $P_\sigma(q^*) \geq P(q^*) = q^*$ and that $(I - P'_\sigma(x))^{-1}$ exists and is non-negative. Thus the conditions of Lemma 3.18 hold, and we can conclude that $q^* - \mathcal{N}_\sigma(x) \leq \frac{\lambda}{2}(1 - q^*)$. □

Proposition 3.10 for minPPSs follows from Lemmas 3.17 and 3.19.

3.5 Putting it Together

In this subsection we use the properties shown in the previous subsections to analyze the algorithm and show Theorem 3.3 and Corollary 3.4. We show first that all iterations are well-defined.

Lemma 3.20. *If we run the rounded-down-GNM starting with $x^{(0)} := 0$ on a max/minPPS, $x = P(x)$, with LFP q^*, $0 < q^* < 1$, then for all $k \geq 0$, $x^{(k)}$ is well-defined and $0 \leq x^{(k)} \leq q^*$.*

Proof. The base case $x^{(0)} = 0$ is immediate for both claims.

For the induction step, suppose the claims hold for k and thus $0 \leq x^{(k)} \leq q^*$. From Proposition 3.10, $I(x^{(k)})$ is well-defined and $I(x^{(k)}) \leq q^*$. Furthermore, since $x^{(k+1)}$ is obtained from $I(x^{(k)})$ by rounding down all coordinates, except setting to 0 any that are negative, and since obviously $q^* > 0$, we have that $0 \leq x^{(k+1)} \leq q^*$. □

We analyze now the running time.

In [13] we gave a polynomial time algorithm, in the standard Turing model of computation, for approximating the LFP of a PPS, $x = P(x)$, using Newton's method. The proof in [13] uses induction based on the "halving lemma", Lemma 2.11. We of course now have suitable "halving lemmas" for maxPPSs and minPPSs, namely, Lemmas 3.13 and 3.19. In [13], the following bound was used for the base case of the induction:

Lemma 3.21 (Theorem 3.14 from [13])**.** *If $0 < q^* < 1$ is the LFP of a PPS, $x = P(x)$, in n variables, then for all $i \in \{1, \ldots, n\}$:*

$$1 - q_i^* \geq 2^{-4|P|}$$

In other words, $0 < q_i^* \leq 1 - 2^{-4|P|}$, for all $i \in \{1, \ldots, n\}$.

We can easily derive from this an analogous Lemma for the setting of max/minPPSs:

Lemma 3.22. *If $0 < q^* < 1$ is the LFP of a max/minPPS, $x = P(x)$, in n variables, then for all $i \in \{1, \ldots, n\}$:*

$$1 - q_i^* \geq 2^{-4|P|}$$

In other words, $0 < q_i^* \leq 1 - 2^{-4|P|}$, for all $i \in \{1, \ldots, n\}$.

Proof. Let τ be any optimal policy for $x = P(x)$. We know it exists, by Theorem 2.5. Lemma 3.21 gives that $1 - q_i^* \geq 2^{-4|P_\tau|}$. All we need to note is that $|P| \geq |P_\tau|$, which clearly holds using any sensible encoding for P and P_τ, in the sense that we should need no more bits needed to encode $x_i = x_j$ than to encode $x_i = \max\{x_j, x_k\}$ or $x_i = \min\{x_j, x_k\}$. □

For a vector $v > 0$, we will use the notation $v_{\min}$ to denote its minimum entry. Thus, the lemma says that if $q^* < 1$ then $(1 - q^*)_{\min} \geq 2^{-4|P|}$.

We bound now the distance of the iterates $x^{(k)}$ of GNM from the LFP q^*.

Lemma 3.23. *For a max/minPPS, $x = P(x)$, with LFP q^*, such that $0 < q^* < 1$, if we apply rounded-down-GNM with parameter h, starting at $x^{(0)} := 0$, then for all $k \geq 0$, we have:*

$$\|q^* - x^{(k)}\|_\infty \leq (2^{-k} + 2^{-h+1})2^{4|P|}$$

Proof. Since $x^{(0)} := 0$:

$$q^* - x^{(0)} = q^* \leq 1 \leq \frac{1}{(1-q^*)_{\min}}(1-q^*) \tag{5}$$

For any $k > 0$, if $q^* - x^{(k-1)} \leq \lambda(1-q^*)$, then by Proposition 3.10 (which was proved separately for maxPPSs and minPPSs, in Lemmas 3.13 and 3.19, respectively), we have:

$$q^* - I(x^{(k-1)}) \leq (\frac{\lambda}{2})(1-q^*) \tag{6}$$

Observe that after every iteration $k > 0$, in every coordinate i we have:

$$x_i^{(k)} \geq I(x^{(k-1)})_i - 2^{-h} \tag{7}$$

This holds simply because we are rounding down $I(x^{(k-1)})_i$ by at most 2^{-h}, unless it is negative in which case $x_i^{(k)} = 0 > I(x^{(k-1)})_i$. Combining the two inequalities (6) and (7) yields the following inequality:

$$q^* - x^{(k)} \leq (\frac{\lambda}{2})(1-q^*) + 2^{-h}1 \leq (\frac{\lambda}{2} + \frac{2^{-h}}{(1-q^*)_{\min}})(1-q^*)$$

Taking inequality (5) as the base case (with $\lambda = \frac{1}{(1-q^*)_{\min}}$), it follows by induction on k, for all $k \geq 0$:

$$q^* - x^{(k)} \leq (2^{-k} + \sum_{i=0}^{k-1} 2^{-(h+i)})\frac{1}{(1-q^*)_{\min}}(1-q^*)$$

But $\sum_{i=0}^{k-1} 2^{-(h+i)} \leq 2^{-h+1}$ and $\frac{\|1-q^*\|_\infty}{(1-q^*)_{\min}} \leq \frac{1}{(1-q^*)_{\min}} \leq 2^{4|P|}$, by Lemma 3.22. Thus:

$$q^* - x^{(k)} \leq (2^{-k} + 2^{-h+1})2^{4|P|}$$

Clearly, we have $q^* - x^{(k)} \geq 0$ for all k. Thus we have shown that for all $k \geq 0$:

$$\|q^* - x^{(k)}\|_\infty \leq (2^{-k} + 2^{-h+1})2^{4|P|}.$$

□

Combining Lemmas 3.20 and 3.23, we can prove Theorem 3.3.

Proof of Theorem 3.3. In Lemma 3.23 let $k := j + 4|P| + 1$ and $h := j + 2 + 4|P|$. We have: $\|q^* - x^{(j+1+4|P|)}\|_\infty \leq 2^{-(j+1)} + 2^{-(j+1)} = 2^{-j}$. □

Corollary 3.4 follows readily:

Proof of Corollary 3.4. First, we use the algorithms given in [19] (Theorems 11 and 13), or the faster algorithms in the Appendix of this paper, to identify those variables x_i with $q_i^* = 0$ or $q_i^* = 1$ in time polynomial in $|P|$. Then we remove these variables from the max/minPPS by substituting their known values into the equations for other variables. This gives us a max/minPPS with LFP $0 < q^* < 1$ and does not increase $|P|$. Then we use the iterated GNM, with rounding down, as outlined earlier in this section. In each iteration of GNM we solve an LP. Each LP has at most

$n \leq |P|$ variables, at most $2n$ constraints and the numerators and denominators of each rational coefficient are no larger than $2^{j+2+4|P|}$, so it can be solved in time polynomial in $|P|$ and j using standard algorithms. We need only $1+2+4|P|$ iterations involving one LP each. Putting back the removed 0 and 1 values into the resulting vector gives us the full result q^*. This can all be done in polynomial time. □

4 Computing an ϵ-optimal policy in P-time

First let us note that we can not hope to compute an optimal policy in P-time, without a major breakthrough:

Theorem 4.1. *Computing an optimal policy for a max/minPPS is* PosSLP*-hard.*

Proof. Recall from [18, 13] that the termination (extinction) probability vector q^* of a Branching Process (or of a 1-exit Recursive Markov Chain (1-RMC)) can be equivalently viewed as the LFP of a purely probabilistic PPS, and vice-versa.

It was shown in [18] (Theorem 5.3), that given a PPS (or equivalently, a BP or 1-RMC), and given a rational probability p, it is PosSLP-hard to decide whether the LFP $q_1^* > p$, for a given rational p, as well as to decide whether $q_1^* < p$. (In fact, these hardness results hold already even if $p = 1/2$.)

The fact that computing an optimal policy for max/minPPSs is PosSLP-hard follows easily from this: For the case of maxPPSs (minPPS, respectively), given a PPS, $x = P(x)$, and given p, we simply add a new variable x_0 to the PPS, and a corresponding equation:

$$x_0 = \max\{p, x_1\} \quad (\text{resp. } , x_0 = \min\{p, x_1\}) \tag{8}$$

It is clear that $q_1^* > p$ ($q_1^* < p$, respectively) holds for the original PPS if and only if in any optimal policy σ, for the augmented maxPPS (minPPS, respectively), the policy picks x_1 rather than p on the RHS of equation (8). So, if we could compute an optimal policy for a maxPPS (minPPS), we would be able to decide whether $q_1^* > p$ (whether $q_1^* < p$, respectively). □

Since we can not hope to compute an optimal policy for max/minPPSs in P-time without a major breakthrough, we will instead seek to find a policy σ such that $\|q_\sigma^* - q^*\|_\infty \leq \epsilon$ for a given desired $\epsilon > 0$, in time *poly*($|P|$, $\log(1/\epsilon)$). We have an algorithm for approximating q^*. Can we use a sufficiently close approximation, q, to q^* to find such an ϵ-optimal strategy? Once we have an approximation q, it seems natural to consider policies σ such that $P_\sigma(q) = P(q)$. For minPPSs, this means choosing for each type-M variable x_i with equation of the form $x_i = \min\{x_j, x_k\}$, the variable x_j or x_k that has the lowest value in the approximate vector q and for maxPPSs choosing the variable that has the highest value in q. It turns out that this works for minPPSs (provided that q is sufficiently close to q^*), while for maxPPSs we need to select the policy σ more carefully.

Before getting into the details, we outline the basic approach for the algorithm and the proof. For most of this section we focus on the case when the LFP q^* satisfies $0 < q^* < 1$; at the end of the section we will extend the policy to the variables that have value 0 or 1 in the LFP. We compute a sufficiently close approximation q of the LFP q^* of the given max/minPPS $x = P(x)$, and let σ be a policy such that $P_\sigma(q) = P(q)$. We would like to show that the corresponding LFP q_σ^* of the PPS $x = P_\sigma(x)$ is within distance ϵ of the LFP q^* of the given max/minPPS. We know that q is close to q^*, hence it suffices to show that q is sufficiently close also to q_σ^*. Toward this purpose, in the first

part of the proof, we bound the distance $\|q^*_\sigma - q\|_\infty$ by the norm of $(I - P'_\sigma(x))^{-1}$ for a certain value of x and $\|q^* - q\|_\infty$ (Lemma 4.4 below). In the second part of the proof we then use a result from [13] for PPS in order to bound the norm of the matrix $(I - P'_\sigma(x))^{-1}$. For minPPS we show that the hypothesis of this result of [13] is satisfied by any policy σ such that $P_\sigma(q) = P(q)$, if q is close enough to q^*. For maxPPS more effort is required and we give an algorithm that chooses carefully a policy σ so that the hypothesis is satisfied.

We start with a lemma on PPS, which will then be applied in our case to the PPS $x = P_\sigma(x)$ for an appropriate policy σ. The proof is given in the appendix.

Lemma 4.2. *If $x = P(x)$ is a PPS, with LFP q^*, and the matrix $(I - P'(\frac{1}{2}(q^* + y)))^{-1}$ exists, then:*

$$q^* - y = (I - P'(\frac{1}{2}(q^* + y)))^{-1}(P(y) - y) \tag{9}$$

The norm of the left-hand side $\|q^* - y\|$ of the equation (9) of Lemma 4.2 is bounded by the product of the norms of the matrix and the vector $P(y) - y$ on the right-hand side. We can bound the norm of $P(y) - y$ for a PPS, and more generally for a max/minPPS, in terms of the distance of y from the LFP (see the Appendix for the proof).

Lemma 4.3. *If $x = P(x)$ is a max/minPPS with LFP q^*, and if $0 \leq y \leq q^*$, then $\|P(y) - y\|_\infty \leq 2\|q^* - y\|_\infty$.*

From the previous two lemmas we can derive the bound in the following lemma (see the Appendix for the proof). For a square matrix A, $\rho(A)$ denotes its spectral radius. A basic property is that, if A is a non-negative matrix and $\rho(A) < 1$, then the matrix $I - A$ is nonsingular and $(I - A)^{-1} = \sum_{k=0}^{\infty} A^k$ is non-negative (see e.g. [23]).

Lemma 4.4. *For a max/minPPS, $x = P(x)$, given $0 \leq q \leq q^*$, such that $q < 1$, and a policy σ such that $P(q) = P_\sigma(q)$, and such that $\rho(P'_\sigma(\frac{1}{2}(q^* + q^*_\sigma))) < 1$, and thus $(I - P'_\sigma(\frac{1}{2}(q^* + q^*_\sigma)))^{-1}$ exists and is non-negative, then*

$$\|q^*_\sigma - q^*\|_\infty \leq (2\|(I - P'_\sigma(\frac{1}{2}(q^*_\sigma + q^*)))^{-1}\|_\infty + 1)\|q^* - q\|_\infty$$

To apply Lemma 4.4 we need to show the existence of the matrix $(I - P'_\sigma(\frac{1}{2}(q^*_\sigma + q^*)))^{-1}$ and bound its norm. For this, we use the following fact for PPS, which is proved in [13].

Lemma 4.5. *([13], Theorem 5.1) If $x = P(x)$ is a PPS with LFP $q^* > 0$ then*
(i) If $q^* < 1$ and $0 \leq y < 1$, then $\rho(P'(\frac{1}{2}(y + q^*))) < 1$, thus $(I - P'(\frac{1}{2}(y + q^*)))^{-1}$ exists and is non-negative, and

$$\|(I - P'(\frac{1}{2}(y + q^*)))^{-1}\|_\infty \leq 2^{10|P|}\max\{2(1 - y)^{-1}_{\min}, 2^{|P|}\}$$

(ii) If $q^* = 1$ and $x = P(x)$ is strongly connected (i.e. every variable depends directly or indirectly on every other) and $0 \leq y < 1 = q^*$, then $\rho(P'(y)) < 1$, thus $(I - P'(y))^{-1}$ exists and is non-negative, and

$$\|(I - P'(y))^{-1}\|_\infty \leq 2^{4|P|}\frac{1}{(1 - y)_{\min}}$$

To apply Lemma 4.5(i) on the PPS $x = P_\sigma(x)$ and complete the proof has some complications due to the following fact: Although we assume that $0 < q^* < 1$, it need not be true for an arbitrary policy σ that $0 < q^*_\sigma < 1$.

Example 4.1. Consider the following maxPPS $x = P(x)$:

$$x_1 = \max(x_2, x_4);\ x_2 = \max(x_1, x_3);\ x_3 = \max(x_2, x_5);\ x_4 = 0.25x_3 + 0.5x_5 + 0.25;\ \ x_5 = x_1x_4;$$

The LFP is $q^* = (0.5, 0.5, 0.5, 0.5, 0.25)$, and it is achieved by the optimal policy τ which selects $\tau(x_1) = x_4, \tau(x_2) = x_1, \tau(x_3) = x_2$. Consider however the policy σ which selects $\sigma(x_1) = x_2, \sigma(x_2) = x_3, \sigma(x_3) = x_2$. The induced PPS $x = P_\sigma(x)$ is:

$$x_1 = x_2;\ \ x_2 = x_3;\ \ x_3 = x_2;\ \ x_4 = 0.25x_3 + 0.5x_5 + 0.25;\ \ x_5 = x_1x_4$$

Note that $P_\sigma(q^*) = q^*$. However, the LFP of the PPS $x = P_\sigma(x)$ is $q^*_\sigma = (0, 0, 0, 0.25, 0)$. □

But the following obviously does hold:

Proposition 4.6. *Given a max/minPPS, $x = P(x)$, with LFP q^* such that $0 < q^* < 1$, for any policy σ:*
(i) If $x = P(x)$ is a maxPPS then $q^*_\sigma < 1$.
(ii) If $x = P(x)$ is a minPPS, then $q^*_\sigma > 0$.

Proof. If $x = P(x)$ is a maxPPS, then clearly $q^*_\sigma \leq q^* < 1$, because σ can be no better than an optimal strategy. Likewise, if $x = P(x)$ is a minPPS, then $0 < q^* \leq q^*_\sigma$, for the same reason. □

For maxPPSs, we may have that some coordinate of q^*_σ is equal to 0 and for minPPSs we may have that some coordinate of q^*_σ is equal to 1, even when $0 < q^* < 1$. This is the source of different complications for the max and the min case, and we give separate proofs for the two cases.

MinPPS

For minPPSs we shall show that if y is a sufficiently close approximation to q^*, then any policy σ with $P(y) = P_\sigma(y)$ is ϵ-optimal. The maxPPS case will not be so simple: the analogous statement is false for maxPPSs.

Theorem 4.7. *If $x = P(x)$ is a minPPS, with LFP $0 < q^* < 1$, and $0 \leq \epsilon \leq 1$, and $0 \leq y \leq q^*$, such that $\|q^* - y\|_\infty \leq 2^{-14|P|-3}\epsilon$, then for any policy σ with $P_\sigma(y) = P(y)$, $\|q^* - q^*_\sigma\|_\infty \leq \epsilon$.*

Proof. By Proposition 4.6, $q^*_\sigma \geq q^*$, and so $q^*_\sigma > 0$. Suppose for now that $q^*_\sigma < 1$ (we will show this later). Then applying Lemma 4.5 (i), for the case where we set $y := q^*$ and the PPS is $x = P_\sigma(x)$, yields that

$$\|(I - P'_\sigma(\frac{1}{2}(q^* + q^*_\sigma)))^{-1}\|_\infty \leq 2^{10|P_\sigma|}\max\{\frac{2}{(1-q^*)_{\min}}, 2^{|P_\sigma|}\}$$

Note that $|P_\sigma| \leq |P|$. From Lemma 3.22, $(1 - q^*)_{\min} \geq 2^{-4|P|}$. Thus

$$\|(I - P'_\sigma(\frac{1}{2}(q^* + q^*_\sigma)))^{-1}\|_\infty \leq 2^{14|P|+1}$$

Lemma 4.4 now gives that

$$\|q^* - q^*_\sigma\|_\infty \leq (2^{14|P|+2} + 1)\|q^* - y\|_\infty \leq \epsilon$$

Thus, under the assumption that $q^*_\sigma < 1$, we are done.

To complete the proof, we now show that $q^*_\sigma < 1$. Suppose, for a contradiction, that for some i, $(q^*_\sigma)_i = 1$. Then by results in [18], $x = P_\sigma(x)$ (i.e., its dependency graph) has a bottom strongly connected component S with $q^*_S = 1$. If x_i is in S then only variables in S appear in $(P_\sigma)_i(x)$, so we write $x_S = P_S(x)$ for the PPS which is formed by such equations. We also have that $P'_S(1)$ is irreducible and that the least fixed point solution of $x_S = P_S(x_S)$ is $q^*_S = 1$. Take y_S to be the subvector of y with coordinates in S. Now if we apply Lemma 4.5 (ii) to the PPS $x_S = P_S(x)$, by taking the y in its statement to be $\frac{1}{2}(y_S + 1)$, it gives that

$$\|(I - P'_S(\frac{1}{2}(y_S + 1)))^{-1}\|_\infty \leq 2^{4|P_S|}\frac{1}{\frac{1}{2}(1 - y_S)_{\min}}$$

But $|P_S| \leq |P|$ and $(1 - y_S)_{\min} \geq (1 - q^*)_{\min} \geq 2^{-4|P|}$. Thus

$$\|(I - P'_S(\frac{1}{2}(y_S + 1)))^{-1}\|_\infty \leq 2^{8|P|+1}$$

Lemma 4.2 gives that

$$1 - y_S = (I - P'_S(\frac{1}{2}(1 + y_S)))^{-1}(P_S(y_S) - y_S)$$

Taking norms and re-arranging gives:

$$\|P_S(y_S) - y_S)\|_\infty \geq \frac{\|1 - y_S\|_\infty}{\|(I - P'_S(\frac{1}{2}(y_S + 1)))^{-1}\|_\infty} \geq \frac{2^{-4|P|}}{2^{8|P|+1}} \geq 2^{-12|P|-1}$$

However $\|P_S(y_S) - y_S)\|_\infty \leq \|P_\sigma(y) - y\|_\infty$ and $P_\sigma(y) = P(y)$. We deduce that $\|P(y) - y\|_\infty \geq 2^{-12|P|-1}$. Lemma 4.3 states that $\|P(y) - y\|_\infty \leq 2\|q^* - y\|_\infty$. We thus have $\|q^* - y\|_\infty \geq 2^{-12|P|-2}$. This contradicts our assumption that $\|q^* - y\|_\infty \leq 2^{-14|P|-3}\epsilon$ for some $\epsilon \leq 1$. □

MaxPPS

Now we proceed to the harder case of maxPPSs. The main theorem in this case is the following.

Theorem 4.8. *If $x = P(x)$ is a maxPPS with $0 < q^* < 1$ and given $0 \leq \epsilon \leq 1$ and a vector y, with $0 \leq y \leq q^*$, such that $\|q^* - y\|_\infty \leq 2^{-14|P|-3}\epsilon$, then we can compute a policy σ such that $\|q^* - q^*_\sigma\|_\infty \leq \epsilon$ in time polynomial in $|P|$ and $\log(1/\epsilon)$.*

We need a policy σ such that we can apply Lemma 4.5 to $x = P_\sigma(x)$, and for which we can get good bounds on $\|P_\sigma(y) - y\|_\infty$. Firstly we show that such policies exist. In fact, any optimal policy will do: for an optimal policy τ, $q^*_\tau > 0$ and Lemma 4.3 applied to $x = P_\tau(x)$ gives that $\|P_\tau(y) - y\|_\infty \leq 2^{-14|P|-2}\epsilon$. Unfortunately the optimal policy might be hard to find (Theorem 4.1). Furthermore, if we select any policy σ such that $P(y) = P_\sigma(y)$, it is possible that the corresponding LFP q^*_σ of the PPS $x = P_\sigma(x)$ has some coordinates equal to 0, and thus we cannot apply directly Lemma 4.5. To prove the theorem, we will give below an algorithm that computes a suitable policy σ such that $q^*_\sigma > 0$.

First, note that given a policy σ and the PPS $x = P_\sigma(x)$, we can easily test in polynomial time whether the LFP $q^*_\sigma > 0$ (see, e.g., Theorem 2.2 of [18]). We shall make use of the following easy fact, shown in the Appendix:

Lemma 4.9. *If $x = P(x)$ is a PPS with n variables, and with LFP q^*, then for any variable index $i \in \{1, \ldots, n\}$ the following are equivalent:*
(i) $q^*_i > 0$.
(ii) there is a $k > 0$ such that $(P^k(0))_i > 0$.
(iii) $(P^n(0))_i > 0$.

Given the maxPPS, $x = P(x)$, with $0 < q^* < 1$, and given a vector y that satisfies the conditions of Theorem 4.8, we shall use the following algorithm to obtain the policy we need:

1. Initialize the policy σ to any policy such that $P_\sigma(y) = P(y)$.
2. Calculate for which variables x_i in $x = P_\sigma(x)$ we have $(q^*_\sigma)_i = 0$. Let S_0 denote this set of variables. (We can do this in P-time; see e.g., Theorem 2.2 of [18].)
3. If for all i we have $(q^*_\sigma)_i > 0$, i.e., if $S_0 = \emptyset$, then terminate and output the policy σ.
4. Otherwise, look for a variable x_i, where $P_i(x)$ is of form M, with $P_i(x) = \max\{x_j, x_k\}$, and where $(q^*_\sigma)_i = 0$ but one of x_j, x_k, say x_j, has $(q^*_\sigma)_j > 0$ and where furthermore $\|y_i - y_j\| \leq 2^{-14|P|-2}\epsilon$. (We shall establish that such a pair x_i and x_j will always exist when we are at this step of the algorithm.)

   Let σ' be the policy that chooses x_j at x_i but is otherwise identical to σ. Set $\sigma := \sigma'$ and return to step 2.

Example 4.2. Consider the maxPPS of Example 4.1.

$$x_1 = \max(x_2, x_4);\ x_2 = \max(x_1, x_3);\ x_3 = \max(x_2, x_5);\ x_4 = 0.25x_3 + 0.5x_5 + 0.25;\quad x_5 = x_1x_4$$

Let y be a sufficiently close approximation to the LFP $q^* = (0.5, 0.5, 0.5, 0.5, 0.25)$ and suppose that $y_2 > y_4$ and $y_3 > y_1$. The policy σ that satisfies $P_\sigma(y) = P(y)$ selects $\sigma(x_1) = x_2, \sigma(x_2) = x_3, \sigma(x_3) = x_2$. As in Example 4.1, the LFP of the PPS $x = P_\sigma(x)$ is $q^*_\sigma = (0, 0, 0, 0.25, 0)$; the algorithm will only compute the set $S_0 = \{x_1, x_2, x_3, x_5\}$ of variables with value 0 in q^*_σ, not q^*_σ itself. In Step 4 the algorithm will switch the choice for variable x_1 since $(q^*_\sigma)_4 > 0$ and $y_1 \approx y_4 \approx 0.5$, and will set $\sigma'(x_1) = x_4$. The new induced PPS $x = P_{\sigma'}(x)$ is

$$x_1 = x_4;\ x_2 = x_3;\ x_3 = x_2;\ x_4 = 0.25x_3 + 0.5x_5 + 0.25;\quad x_5 = x_1x_4$$

It has LFP $q^*_{\sigma'} \approx (0.29, 0, 0, 0.29, 0.085)$ and the new $S_0 = \{x_2, x_3\}$. Even though x_3 has a successor, x_5, with $(q^*_{\sigma'})_5 > 0$, the algorithm will not switch the choice for x_3 because $y_5 \approx q^*_5 = 0.25 \ll y_3 \approx 0.5$. Rather, it will switch the choice for variable x_2 and will set $\sigma''(x_2) = x_1$, since $(q^*_{\sigma'})_1 > 0$ and $y_1 \approx 0.5 \approx y_2$. The new induced PPS $x = P_{\sigma''}(x)$ is

$$x_1 = x_4;\ x_2 = x_1;\ x_3 = x_2;\ x_4 = 0.25x_3 + 0.5x_5 + 0.25;\quad x_5 = x_1x_4$$

and has LFP $q^*_{\sigma''} = (0.5, 0.5, 0.5, 0.5, 0.25)$, thus the new $S_0 = \emptyset$. So the algorithm will terminate and output σ'', which in this case is the optimal policy. □

Lemma 4.10. *The steps of the above algorithm are always well-defined, and the algorithm always terminates in at most n iterations with a policy σ such that $q^*_\sigma > 0$ and $\|P_\sigma(y) - y\|_\infty \leq 2^{-14|P|-2}\epsilon$.*

Proof. Firstly, to show that the steps of the algorithm are always well-defined, we need to show that if q^*_σ has some coordinate equal to 0, then step 4 will find a pair of variables x_i, x_j that satisfy the condition of step 4 to switch the policy at x_i.

Suppose that q^*_σ has some coordinate equal to 0. Let τ be an optimal policy; then $q^*_\tau = q^* > 0$. So by Lemma 4.9, $P^n_\tau(0) > 0$. For any variable x_j with $(P_\tau(0))_j > 0$, the equation $x_j = P_j(x)$ must have form L and not M so $(P_\sigma(0))_j > 0$ and so $(q^*_\sigma)_j > 0$. There must be a least k, $k_{\min}$ with $1 < k_{\min} \leq n$, such that there is a variable x_j with $(P^k_\tau(0))_j > 0$ but $(q^*_\sigma)_j = 0$. Let x_i be a variable such that $(P^{k_{\min}}_\tau(0))_i > 0$ but $(q^*_\sigma)_i = 0$. We claim that x_i is of type M.

Suppose that $x_i = P_i(x)$ has form Q. Then $P_i(x) = x_j x_l$ for some variables x_j, x_l. We have $0 < (P^{k_{\min}}_\tau(0))_i = (P^{k_{\min}-1}_\tau(0))_j (P^{k_{\min}-1}_\tau(0))_l$. So $(P^{k_{\min}-1}_\tau(0))_j > 0$ and $(P^{k_{\min}-1}_\tau(0))_l > 0$. The minimality of $k_{\min}$ now gives us that $(q^*_\sigma)_j > 0$ and $(q^*_\sigma)_l > 0$. So $(q^*_\sigma)_i = (q^*_\sigma)_j (q^*_\sigma)_l > 0$. This is a contradiction. Thus, $x_i = P_i(x)$ does not have form Q.

Similarly, $x_i = P_i(x)$ does not have form L. So $x_i = P_i(x)$ has form M. There are variables x_j, x_l with $P_i(x) = \max\{x_j, x_l\}$. Suppose, w.l.o.g. that $(P_\tau(x))_i = x_j$. We have $P^{k_{\min}}_\tau(0))_i > 0$ and so $(P^{k_{\min}-1}(0))_j > 0$. By minimality of $k_{\min}$, we have that $(q^*_\sigma)_j > 0$. We have that $(q^*_\sigma)_i = 0$ and so $(P_\sigma(x))_i = x_l$.

Lemma 4.3 applied to the system $x = P_\tau(x)$ gives that $\|P_\tau(y) - y\|_\infty \leq 2^{-14|P|-2}\epsilon$. So $|y_i - y_j| = |y_i - (P_\tau(y))_i| \leq 2^{-14|P|-2}\epsilon$. Thus, step 4 could use x_i and change the policy σ at x_i (i.e., switch $\sigma(i)$) from x_l to x_j.

Next, we need to show that the algorithm terminates:

Claim 4.11. *If step 4 switches the variable x_i with $P_i(x) = \max\{x_j, x_k\}$ from $(P_\sigma(x))_i = x_k$ to $(P_{\sigma'}(x))_i = x_j$, then*
(i) $q^*_{\sigma'} \geq q^*_\sigma$,
(ii) $(q^*_{\sigma'})_i > 0$,
(iii) The set of variables x_l with $(q^*_{\sigma'})_l > 0$ is a strict superset of the set of variables x_l with $(q^*_\sigma)_l > 0$.

Proof. Recall that step 4 will only switch if $(q^*_\sigma)_i = 0$ and $(q^*_\sigma)_j > 0$.

(i) We show that, for any $t > 0$, $P^t_{\sigma'}(0) \geq P^t_\sigma(0)$. We use induction on t.
The base case $t = 1$, is clear, because the only indices i where $P_i(0) \neq 0$ are when $P_i(0)$ has form L, in which case $P_i(0) = (P_{\sigma'}(0))_i = (P_\sigma(0))_i$.

For the inductive case: note firstly that $P_\sigma(x)$ and $P_{\sigma'}(x)$ only differ on the ith coordinate. $(q^*_\sigma)_i = 0$, so for any t, $(P^t_\sigma(0))_i = 0$. Suppose that $P^t_{\sigma'}(0) \geq P^t_\sigma(0)$. Then by monotonicity $P^{t+1}_{\sigma'}(0) \geq P_{\sigma'}(P^t_\sigma(0))$. But $(P_{\sigma'}(P^t_\sigma(0)))_r = (P^{t+1}_\sigma(0))_r$ when $r \neq i$. Furthermore, $(P_{\sigma'}(P^t_\sigma(0)))_i \geq 0 = (P^{t+1}_\sigma(0))_i$. So $P_{\sigma'}(P^k_\sigma(0)) \geq P^{k+1}_\sigma(0)$. We thus have that $P^{t+1}_{\sigma'}(0) \geq P^{t+1}_\sigma(0)$.

We know that as $t \to \infty$, $P^t_{\sigma'}(0) \to q^*_{\sigma'}$ and $P^t_\sigma(0) \to q^*_\sigma$. So $q^*_{\sigma'} \geq q^*_\sigma$.

(ii) We have $(q^*_{\sigma'})_i = (q^*_{\sigma'})_j$. By (i) $(q^*_{\sigma'})_j \geq (q^*_\sigma)_j$. We chose x_j such that $(q^*_\sigma)_j > 0$. So $(q^*_{\sigma'})_i > 0$.

(iii) If $(q^*_\sigma)_l > 0$, then by (i) $(q^*_{\sigma'})_l > 0$. Also $(q^*_\sigma)_i = 0$ and by (ii) $(q^*_{\sigma'})_i > 0$.

□

Thus, if at some stage of the algorithm we do not yet have $q^*_\sigma > 0$, then step 4 always gives us a new σ' with more coordinates having $(q^*_{\sigma'})_i > 0$. This can happen at most n times. Furthermore,

note that if $\|P_\sigma(y) - y\|_\infty \leq 2^{-14|P|-2}\epsilon$ then $\|P_{\sigma'}(y) - y\|_\infty \leq 2^{-14|P|-2}\epsilon$. Our starting policy has $\|P_\sigma(y) - y\|_\infty = \|P(y) - y\|_\infty \leq 2^{-14|P|-2}\epsilon$. The algorithm terminates in at most n iterations and gives a σ with $q^*_\sigma > 0$ and $\|P_\sigma(y) - y\|_\infty \leq 2^{-14|P|-2}\epsilon$. □

We can now complete the proof of Theorem 4.8:

Proof of Theorem 4.8. Using the algorithm, we find a σ with $\|y - P_\sigma(y)\|_\infty \leq 2^{-14|P|-2}\epsilon$ and $q^*_\sigma > 0$. By Proposition 4.6, $q^*_\sigma < 1$. Also, $y < 1$ since $y \leq q^*$ and $q^* < 1$. Applying Lemma 4.5 (i) to the PPS $x = P_\sigma(x)$ and point y gives that $(I - P'_\sigma(\frac{1}{2}(y + q^*_\sigma)))^{-1}$ exists and

$$\|(I - P'_\sigma(\frac{1}{2}(y + q^*_\sigma)))^{-1}\|_\infty \leq 2^{10|P_\sigma|}\max\{\frac{2}{(1-y)_{\min}}, 2^{|P_\sigma|}\}$$

We have $|P_\sigma| \leq |P|$. From the fact that there always exists an optimal policy and from Lemma 3.22 (Theorem 3.14 of [13]), it follows that $(1 - q^*)_{\min} \geq 2^{-4|P|}$, and since $y \leq q^*$, we have $(1-y)_{\min} \geq 2^{-4|P|}$. So

$$\|(I - P'_\sigma(\frac{1}{2}(y + q^*_\sigma)))^{-1}\|_\infty \leq 2^{14|P|+1} \tag{10}$$

We can not use Lemma 4.4 as stated because we need not have $P(y) = P_\sigma(y)$. We will use Lemma 4.2 instead. As observed above, the matrix $(I - P'_\sigma(\frac{1}{2}(y + q^*_\sigma)))^{-1}$ exists. Applying Lemma 4.2 to the PPS $x = P_\sigma(x)$, and taking norms, we get the inequality

$$\|q^*_\sigma - y\|_\infty \leq \|(I - P'_\sigma(\frac{1}{2}(q^*_\sigma + y)))^{-1}\|_\infty \|P_\sigma(y) - y\|_\infty \tag{11}$$

From Lemma 4.10 we have

$$\|P_\sigma(y) - y\|_\infty \leq 2^{-14|P|-2}\epsilon \tag{12}$$

Combining (10), (12) and (11) yields:

$$\|q^*_\sigma - y\|_\infty \leq \frac{1}{2}\epsilon$$

so $\|q^*_\sigma - q^*\|_\infty \leq \|q^*_\sigma - y\|_\infty + \|q^* - y\|_\infty \leq \frac{1}{2}\epsilon + 2^{-14|P|-3}\epsilon \leq \epsilon$. □

We can extend the policy to the variables that have value 0 or 1 in the LFP and get an ϵ-optimal policy for any maxPPS or min PPS:

Theorem 4.12. *Given a max/minPPS, $x = P(x)$, and given $\epsilon > 0$, we can compute an ϵ-optimal policy for $x = P(x)$ in time poly($|P|$, log$(1/\epsilon)$)*

Proof. First we use the algorithms from [19] or the Appendix to detect variables x_i with $q^*_i = 0$ or $q^*_i = 1$ in time polynomial in $|P|$. Then we can remove these from the max/minPPS by substituting the known values into the equations for other variables. This gives us a max/minPPS with least fixed point $0 < q^* < 1$ and does not increase $|P|$. To use either Theorem 4.8 or Theorem 4.7, it suffices to have a y with $y \leq q^*$ with $q^* - y \leq 2^{-14|P|-3}\epsilon$. Theorem 3.3 says that we can find such a y in time polynomial in $|P|$ and $14|P| - \log(\epsilon)$, which is polynomial in $|P|$ and $\log(1/\epsilon)$ as required. Now depending on whether we have a maxPPS or minPPS, Theorem 4.8 or Theorem 4.7 show that from this y, we can find an ϵ-optimal policy for the max/minPPS with $0 < q^* < 1$ in time polynomial in $|P|$ and $\log(1/\epsilon)$. All that is left to show is that we can extend this policy to the removed variables x_i where $q^*_i = 0$ or $q^*_i = 1$ while still remaining ϵ-optimal.

We next show how this can be done.

For a minPPS, if $q_i^* = 1$ then for any policy σ, $(q_\sigma^*)_i = 1$ so the choice made at such variables x_i is irrelevant. Similarly, for maxPPSs, when $q_i^* = 0$, any choice at x_i is optimal.

For a minPPS with $q_i^* = 0$, if $P_i(x)$ has form M, we can choose any variable x_j with $q_j^* = 0$. There is such a variable: if $P_i(x) = \min\{x_j, x_k\}$ and $q_i^* = 0$ then either $q_j^* = 0$ or $q_k^* = 0$. Let σ be a policy such that for each variable x_i with $q_i^* = 0$, $(q^*)_{\sigma(i)} = 0$. We need to show that $(q_\sigma^*)_i = 0$ for all such variables. Suppose that, for some $k \geq 0$, $(P_\sigma^k(0))_i = 0$ for all x_i such that $q_i^* = 0$. Then $P(P_\sigma^k(0))_i = 0$ for all x_i with $q_i^* = 0$.

To see why this is so, note that whether or not $P_i(z) = 0$ depends only on which coordinates of z are 0, and furthermore if $P_i(z) = 0$ when the set of 0 coordinates of z is S, then for any vector z' where the 0 coordinates of z' are $S' \supseteq S$, we have $P_i(z') = 0$. Since the coordinates S that are 0 in q^* are a subset of the coordinates S' that are 0 in $P_\sigma^k(0)$, and we have $P_i(q^*) = q_i^* = 0$, we thus have $P(P_\sigma^k(0))_i = 0$.

If $P_i(x) = \min\{x_j, x_k\}$ and $q_i^* = 0$ then either $q_j^* = 0$ or $q_k^* = 0$. Suppose w.l.o.g. that $(P_\sigma(x))_i = x_j$. Then $q_j^* = 0$, so by assumption $(P_\sigma^k(0))_j = 0$ and so $(P_\sigma(P_\sigma^k(0)))_i = 0$. We now have enough for $(P_\sigma^{k+1}(0))_i = 0$ for each variable x_i with $q_i^* = 0$. $P_\sigma^0(0) = 0$, so by induction for all $k \geq 0$, $(P_\sigma^k(0))_i = 0$ for all x_i with $q_i^* = 0$. From this, for each variable x_i with $q_i^* = 0$, $(q_\sigma^*)_i = 0$.

The case of a maxPPS that have variables with $q_i^* = 1$ is not so simple. Although it is again the case that if a variable x_i of type M with $P_i(x) = \max\{x_j, x_k\}$ has value $q_i^* = 1$ in the LFP, then at least one of the variables x_j, x_k has also value 1 (i.e. $q_j^* = 1$ or $q_k^* = 1$), but if both variables are 1 in q^*, the policy σ cannot choose arbitrarily one of them for x_i, because then it is possible that in the resulting LFP q_σ^* the variable x_i does not have value 1 (it can get value 0 in fact). Thus, for maxPPS more care is needed to choose the policy for the variables with value $q_i^* = 1$. The P-time algorithm given in [19] to compute the variables with $q_i^* = 1$, produces also a randomized policy for these variables (Lemma 12 in [19]). In Section A of the Appendix we give an improved algorithm that produces a *pure* (non-randomized) policy for these variables (and much faster than the previous algorithm of [19]). □

5 Approximating the value of BSSGs in FNP

In this section we briefly note that, as an easy corollary of our results for BMDPs, we can obtain a TFNP (total NP search problem) upper bound for computing (approximately), the *value* of *Branching simple stochastic games* (BSSG), where the objective of the two players is to maximize, and minimize, the extinction probability. For relevant definitions and background results about these games see [19]. It suffices for our purposes here to point out that, as shown in [19], the value of these games (which are determined) is characterized by the LFP solution of associated min-maxPPSs, $x = P(x)$, where both min and max operators can occur in the equations for different variables. Furthermore, both players have optimal policies (i.e. optimal pure, memoryless strategies) in these games (see [19]).

Corollary 5.1. *Given a max-minPPS, $x = P(x)$, and given a rational $\epsilon > 0$, the problem of approximating the LFP q^* of $x = P(x)$, i.e., computing a vector v such that $\|q^* - v\|_\infty \leq \epsilon$, is in TFNP, as is the problem of computing ϵ-optimal policies for both players. (And thus also, the problem of approximating the value, and computing ϵ-optimal strategies, for BSSGs is in FNP.)*

Proof. Let $x = P(x)$ be the max-minPPS whose LFP, q^*, we wish to compute. First guess pure policies σ and τ for the max and min players, respectively. Then, fix σ as max's strategy, and for the resulting minPPS (with LFP q^*_σ) use our algorithm to compute in P-time an approximate value vector $v_\sigma \leq q^*_\sigma$, such that $\|v_\sigma - q^*_\sigma\|_\infty \leq \epsilon/2$. Next, fix τ as min's strategy, and for the resulting maxPPS (with LFP q^*_τ), use our algorithm to compute in P-time an approximate value vector $v_\tau \leq q^*_\tau$, such that $\|v_\tau - q^*_\tau\|_\infty \leq \epsilon/2$. Finally, check whether $\|v_\sigma - v_\tau\|_\infty \leq \epsilon/2$. If not, then reject this "guess". If so, then output σ and τ as ϵ-optimal policies for max and min, respectively, and output $v := v_\sigma$ as an ϵ-approximation of the LFP, q^*.

We show the correctness of the algorithm. First, we need to show that an output is produced for at least one guess. Indeed, consider the guess where σ, τ are optimal strategies for the two players. Then $q^*_\sigma = q^* = q^*_\tau$, and both v_σ, v_τ are $\leq q^*$ and within $\epsilon/2$ of q^*. Hence $\|v_\sigma - v_\tau\|_\infty \leq \epsilon/2$ and the algorithm will output σ, τ and v_σ.

Second, we need to show that for every guess of the algorithm that results in an output, the output is correct, i.e. σ, τ are ϵ-optimal policies and the value v_σ that is output is within ϵ of q^*. First, note that $q^*_\sigma \leq q^* \leq q^*_\tau$. Since $v_\sigma \leq q^*_\sigma \leq q^*$, we have

$$\begin{aligned} \|q^* - q^*_\sigma\|_\infty \leq \|q^* - v_\sigma\|_\infty &\leq \|q^*_\tau - v_\sigma\|_\infty \\ &\leq \|q^*_\tau - v_\tau\|_\infty + \|v^*_\tau - v_\sigma\|_\infty \\ &\leq \frac{\epsilon}{2} + \frac{\epsilon}{2} = \epsilon \end{aligned}$$

Hence σ is an ϵ-optimal policy for the max player and v_σ is within ϵ of q^*. Since $v_\sigma \leq q^* \leq q^*_\tau$ and $\|q^*_\tau - v_\sigma\|_\infty \leq \epsilon$, it follows also that $\|q^*_\tau - q^*\|_\infty \leq \epsilon$, i.e. τ is an ϵ-optimal policy for the min player. □

It is worth noting that the problem of approximating the value of a BSSG game, to within a desired $\epsilon > 0$, when ϵ is given as part of the input, is already at least as hard as computing the *exact* value of Condon's finite-state simple stochastic games (SSGs) [6], and thus one can not hope for a P-time upper bound without a breakthrough. In fact, it was shown in [19] that even the *qualitative* problem of deciding whether the value $q^*_i = 1$ for a given BSSG (or max-minPPS), which was shown there to be in NP∩coNP, is already at least as hard as Condon's *quantitative* decision problem for finite-state simple stochastic games. (Whereas for finite-state SSGs the qualitative problem of deciding whether the value is 1 is in P-time.)

6 Conclusions

We have provided the first polynomial time algorithms for computing optimal (maximum and minimum) extinction probabilities of Branching MDPs, to arbitrary desired accuracy $\epsilon > 0$, as well as for computing ϵ-optimal policies for extinction. We have done so by providing a P-time algorithm for computing the least fixed point (LFP) solution for systems of probabilistic max/min polynomial Bellman equations (max/minPPSs) to within desired accuracy $\epsilon > 0$. Our algorithms are based on a novel generalization of Newton's method, applied to max/minPPSs.

Extinction probabilities are important quantities for the analysis of multi-type branching processes, and they play a key role in various other analyses of such stochastic processes (see, e.g., [22]). It may thus be expected that efficient algorithms for other analyses of BMDPs may be facilitated by the algorithms we have developed in this paper. Indeed, in a more recent work ([15]) which builds

directly on this paper[5], we have shown that computing optimal *reachability* probabilities for BMDPs can be computed in P-time, to desired precision. (By *reachability* probability in a BMDP we mean the (optimal) probability that, starting from a given population, the population will eventually contain an object of a designated type.) We have done so by showing that optimal *non-reachability* probabilities constitute the *Greatest Fixed Point* (GFP) of (different) max/minPPSs that we can associate with a BMDP, and by showing that a modification of the generalized Newton's method (GNM) developed in this paper can be used to compute the GFP of max/minPPSs, to desired precision, in P-time. It would be interesting to find other classes of infinite-state MDPs, and other systems of max/min polynomial equations (perhaps even some non-monotone ones), where variants of GNM are applicable and yield efficient algorithms.

Finally, our focus in this paper has been on establishing provably polynomial time algorithms for optimal extinction probabilities for BMDPs. Our algorithms are relatively simple to implement, and it will be interesting to empirically evaluate their practical performance. For many MDP models, *value iteration* and *policy iteration* provide practically efficient iterative methods, although their worst-case behavior is (in some cases) known to be poor or is not adequately understood. It is indeed possible, and natural, to consider both value iteration and (suitable approximate versions of) policy iteration for BMDPs by exploiting their max/minPPS Bellman equations. It can be shown that both methods converge to the optimal extinction probabilities for BMDPs. Theoretically, these methods inherit worst-case lower bounds from finite-state MDPs with reachability objectives, as well as worst-case lower bounds which arise already for value iteration for multi-type branching processes [18], as explained in the Introduction. More detailed (theoretical and practical) analysis of the behavior of value and policy iteration methods for BMDPs, and comparison of their practical performance with our P-time algorithms, could be interesting.

References

[1] E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen, and P. B. Miltersen. On the complexity of numerical analysis. *SIAM J. Comput.*, 38(5):1987–2006, 2009.

[2] R. Alur, M. Benedikt, K. Etessami, P. Godefroid, T. Reps, and M. Yannakakis. Analysis of recursive state machines. *ACM Trans. Program. Lang. Syst.*, 27(4), 2005.

[3] A. Berman and R. J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences.* SIAM-Classics in Applied Mathematics, 1994.

[4] T. Brázdil, V. Brozek, K. Etessami, and A. Kucera. Approximating the termination value of one-counter mdps and stochastic games. In *Proc. of 38th ICALP (2)*, pages 332–343, 2011.

[5] T. Brázdil, V. Brozek, V. Forejt, and A. Kucera. Reachability in recursive Markov decision processes. *Inf. Comput.*, 206(5):520–537, 2008.

[6] A. Condon. The complexity of stochastic games. *Inf. & Comp.*, 96(2):203–224, 1992.

[7] C. Courcoubetis and M. Yannakakis. Markov decision processes and regular events. *IEEE Trans. on Automatic Control*, 43(10):1399–1418, 1998.

[5] An extended abstract for this paper appeared earlier in [14].

[8] E. Denardo and U. Rothblum. Totally expanding multiplicative systems. *Linear Algebra Appl.*, 406:142–158, 2005.

[9] J. Esparza, T. Gawlitza, S. Kiefer, and H. Seidl. Approximative methods for monotone systems of min-max-polynomial equations. In *Proc. of 35th ICALP (1)*, pages 698–710, 2008.

[10] J. Esparza, S. Kiefer, and M. Luttenberger. Computing the least fixed point of positive polynomial systems. *SIAM Journal on Computing*, 39(6):2282–2355, 2010.

[11] J. Esparza, A. Kučera, and R. Mayr. Model checking probabilistic pushdown automata. *Logical Methods in Computer Science*, 2(1):1 – 31, 2006.

[12] K. Etessami, A. Stewart, and M. Yannakakis. Polynomial-time algorithms for multi-type branching processes and stochastic context-free grammars. In *Proc. 44th ACM Symposium on Theory of Computing (STOC)*, 2012.

[13] K. Etessami, A. Stewart, and M. Yannakakis. A polynomial-time algorithm for computing extinction probabilities of multi-type branching processes. *SIAM Journal on Computing*, 46(5):1515-1553, 2017.

[14] K. Etessami, A. Stewart, and M. Yannakakis. Polynomial-time algorithms for branching Markov decision processes, and probabilistic min(max) polynomial Bellman equations. In *Proc. 39th Int. Coll. on Automata, Languages and Programming (ICALP)*, 2012.

[15] K. Etessami, A. Stewart, and M. Yannakakis. Greatest fixed points of probabilistic min/max polynomial equations, and reachability for branching Markov decision processes. In *Proc. 42nd Int. Coll. on Automata, Languages and Programming (ICALP)*, 2015. (Full preprint on ArXiv:1502.05533).

[16] K Etessami, D Wojtczak, and M. Yannakakis. Recursive stochastic games with positive rewards. In *Proc. of 35th ICALP (1)*, volume 5125 of *LNCS*, pages 711–723. Springer, 2008. see full tech report at `http://homepages.inf.ed.ac.uk/kousha/bib_index.html`.

[17] K. Etessami, D. Wojtczak, and M. Yannakakis. Quasi-birth-death processes, tree-like QBDs, probabilistic 1-counter automata, and pushdown systems. *Perform. Eval.*, 67(9):837–857, 2010.

[18] K. Etessami and M. Yannakakis. Recursive Markov chains, stochastic grammars, and monotone systems of nonlinear equations. *Journal of the ACM*, 56(1), 66 pages, 2009.

[19] K. Etessami and M. Yannakakis. Recursive Markov decision processes and recursive stochastic games. *Journal of the ACM*, 62 (2), 69 pages, 2015.

[20] E. Feinberg and A. Shwartz (editors). *Handbook of Markov Decision Processes: Methods and Applications.* Kluwer (Springer), 2002.

[21] P. Haccou, P. Jagers, and V. A. Vatutin. *Branching Processes: Variation, Growth, and Extinction of Populations.* Cambridge U. Press, 2005.

[22] T. E. Harris. *The Theory of Branching Processes.* Springer-Verlag, 1963.

[23] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

[24] M. Kimmel and D. E. Axelrod. *Branching processes in biology*. Springer, 2002.

[25] A. N. Kolmogorov and B. A. Sevastyanov. The calculation of final probabilities for branching random processes. *Doklady*, 56:783–786, 1947. (Russian).

[26] C. T. Lam. Implementation of algorithms for analyzing Branching MDPs. *4th Year Undergraduate Thesis Project*, School of Informatics, University of Edinburgh, 2013.

[27] S. Pliska. Optimization of multitype branching processes. *Management Sci.*, 23(2):117–124, 1976/77.

[28] I. Post and Y. Ye. The simplex method is strongly polynomial for deterministic Markov decision processes. *Math. Oper. Res.*, 40(4):859–868, 2015.

[29] M. L. Puterman. *Markov Decision Processes*. Wiley, 1994.

[30] A. Stewart, K. Etessami, and M. Yannakakis. Upper bounds for Newton's method on monotone polynomial systems, and P-time model checking of probabilistic one-counter automata. *Journal of the ACM*, 62(4), 33 pages, 2015.

[31] U. Rothblum and P. Whittle. Growth optimality for branching Markov decision chains. *Math. Oper. Res.*, 7(4):582–601, 1982.

[32] Y. Ye. The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate. *Math. Oper. Res.*, 36(4):593–603, 2011.

Appendix

A Improved algorithms for qualitative analysis of max/minPPSs

In [19] polynomial time algorithms were provided for all *qualitative* decision problems associated with the LFP q^* of a max/minPPS (equivalently, the optimal termination probabilities of a BMDP or an 1-RMDP). Specifically, it was shown that given a maxPPS or a minPPS, $x = P(x)$, we can decide for all variables x_i whether $q_i^* = 1$ (and whether $q_i^* = 0$) in P-time. The algorithm for determining whether $q_i^* = 0$ is easy and runs in linear time, by reducing the problem to AND-OR graph reachability on the dependency graph of variables in $x = P(x)$ (we outline it below).

However, deciding $q_i^* = 1$ is substantially more involved. Unlike the case of checking $q_i^* = 0$, deciding $q_i^* = 1$ depends on the actual coefficients in $x = P(x)$, not just on the "structure" of $x = P(x)$. (By "structure" of $x = P(x)$ we mean simply the sets of monomials with non-zero coefficients on the right hand side of each equation $x_i = P_i(x)$ in $x = P(x)$.)

The algorithms given in [19] for deciding $q_i^* = 1$ are iterative; there are at most n iterations, where n is the number of variables of the max/minPPS $x = P(x)$, and each iteration involves solving a certain linear programming problem. The number of variables of the linear program in the case of minPPSs is $O(n)$ (and its encoding size is linear in that of the given minPPS), but for maxPPSs the LP has $O(n^3)$ variables. (The cubic growth is caused by certain multi-commodity flow variables $f_{i,j,k}$ that were used in those linear programs.)

The upper bounds we establish in this paper for GNM applied to max/minPPSs require preprocessing the max/minPPS to eliminate those variables x_i where $q_i^* = 1$ (and where $q_i^* = 0$),

before applying GNM. Thus, especially in the case of maxPPSs, the large number of variables in the LPs used by the P-time algorithm in [19] for deciding $q_i^* = 1$ make the preprocessing step a bottleneck for practical implementation of these algorithms, including the algorithm we give in this paper for approximating the LFP q^* for max/minPPSs. (Indeed, an attempted implementation of these algorithms ([26]) made clear that checking $q_i^* = 1$ using the algorithms of [19] is a bottleneck.)

In this section, we give significantly improved, more practical, algorithms for deciding $q_i^* = 1$ for max/minPPSs. In particular, for minPPSs we show how to identify all the variables that have value $q_i^* = 1$ in the time required to solve *one* LP with $O(n)$ variables and constraints (and linear encoding size in the size of the minPPS). For maxPPSs our algorithm involves the solution of at most n LPs, but all the LPs have $O(n)$ variables and constraints (and again linear encoding size). We show furthermore how to compute with the same complexity an optimal pure policy for all variables that have value 1.

Before proceeding, let us first recall from [19] that for any max-minPPS, $x = P(x)$, we can easily (in linear time) determine the set Z_0 of variables x_i for which $q_i^* = 0$ in the LFP q^* of $x = P(x)$, using AND-OR graph reachability. In the AND-OR graph reachability problem we are given a directed graph G, and a partition of its set of nodes into three subsets: a subset of type AND nodes, a subset of type OR nodes, and a subset T of *target* nodes. The problem is to compute the (unique) minimal set S of nodes that satisfies the following properties: (i) $T \subseteq S$, (ii) a type AND node v is in S iff all its (immediate) successors are in S, (iii) a type OR node v is in S iff at least one of its (immediate) successors is in S. There is a unique minimal such set S (i.e. every other set S' that satisfies these properties, satisfies $S \subseteq S'$). This is the set of nodes that can *and-or reach* the set T. (The standard graph reachability problem corresponds to the case where there are no AND nodes.) This set S can be computed by a simple iterative algorithm, which initializes S to the set T, and then repeatedly adds any other node v to S if either v is of type AND and all its (immediate) successors are in S, or v is of type OR and some (immediate) successor of v is in S, until no more nodes can be added to S. The algorithm can be implemented to run in linear time in the size of the graph G (its number of nodes and edges).

Suppose a given max-minPPS, $x = P(x)$ in SNF has variables $\{x_1, \ldots, x_n\}$. Let G be the dependency graph of the system $x = P(x)$. In other words, $G = (V, \rightarrow)$ has nodes $V = \{x_1, \ldots, x_n\}$, and it has an edge $x_i \rightarrow x_j$ if and only if x_j appears in $P_i(x)$. Call a type-L variable x_i *leaky* if $P_i(0) = a_{i,0} > 0$ and let $L_{>0}$ denote the set of leaky variables. To determine the set Z_0, we view G as an AND-OR graph. The set T of target nodes is $L_{>0}$. View node x_i of G as an AND node if x_i is either of type Q, or of type M-min (i.e., $P_i(x) = \min(x_j, x_j)$). View node $x_i \notin L_{>0}$ of G as an OR node if x_i is either of type L or of type M-max (i.e., $P_i(x) = \max(x_j, x_k)$). It is then easily seen that the set $L_{>0}$ is AND-OR reachable from x_i in this and-or graph if and only if $q_i^* > 0$ for this max-minPPS. That is, the set Z_0 of variables with value 0 in the LFP q^* is the remaining set of nodes that cannot and-or reach $L_{>0}$ in G.

If the given max-minPPS $x = P(x)$ is in general form (i.e., each $P_i(x) = \max_j\{p_{ij}(x)\}$ or $P_i(x) = \min_j\{p_{ij}(x)\}$ for some probabilistic polynomials p_{ij}), we could transform it to SNF form and apply the above algorithm. Alternatively (and simpler), we can construct the following graph G_r, which we call the *refined graph* of the given (general form) max-minPPS. The graph G_r has one type-M (max or min) node for each variable x_i of the given system, a type-L node for each polynomial p_{ij}, and a type-Q node for each monomial. The graph contains edges from each type-M node x_i to all the type-L nodes p_{ij} in $P_i(x)$, edges from each type-L node p_{ij} to the type-Q nodes corresponding to the monomials of p_{ij}, and edges from each type-Q node to the nodes for

the variables of the monomial. A type-L node corresponding to $p_{ij}(x) = a_{ij,0} + \sum_{r \in R(i,j)} a_{ij,r} x^{\alpha_r}$ is *leaky* if the constant term $a_{ij,0} > 0$. The set Z_0 of variables with value 0 in the LFP q^* is the set of nodes that cannot and-or reach the set $L_{>0}$ of leaky nodes in G_r, where the type Q and type M-max nodes are AND nodes and the type L and type M-min nodes are OR nodes.

A.1 Qualitative classification for minPPSs

We assume without loss of generality that the given minPPS is in simple normal form. At the end of the subsection, we will discuss the case of minPPSs in general form and how the complexity is affected. We first determine the set Z_0 of variables that have value 0 in the LFP and eliminate them from the system. As discussed already, this can be done easily in linear time. So assume henceforth that the LFP q^* of the remaining system $x = P(x)$ satisfies $q^* > 0$. Let G be the dependency graph of the system $x = P(x)$.

We wish to partition the set of (remaining) variables into the sets $Z_1 = \{x_i \mid q_i^* = 1\}$ and $Z_b = \{x_i \mid 0 < q_i^* < 1\}$. We observe first that all the variables in the same strongly connected component (SCC) of G belong to the same set of the partition. To prove this, we need the following simple lemma.

Lemma A.1. *For a minPPS $x = P(x)$, if the vector $y \leq 1$ has $y_j < 1$ and x_j appears in $P_i(x)$, then $P_i(y) < 1$.*

Proof. It is straightforward to verify the conclusion for all three types for variable x_i. For example, for a variable x_i of type M, $P_i(x)$ has the form $P_i(x) = \min(x_j, x_k)$ for some x_k, and we have $P_i(y) = \min(y_j, y_k) \leq y_j < 1$. The cases of type L and Q are similar. □

A consequence of the lemma is that if any variable x_j that appears in $P_i(x)$ has $q_j^* < 1$, then also $q_i^* = P_i(q^*) < 1$. In other words, if the dependency graph G contains an edge $x_i \to x_j$ and $q_j^* < 1$ then also $q_i^* < 1$. By a simple induction it follows that the same property holds if the dependency graph has a path from x_i to x_j.

Corollary A.2. *1. For a minPPS $x = P(x)$, if a variable x_i can reach in the dependency graph a variable x_j with $q_j^* < 1$, then $q_i^* < 1$.*
2. In every strongly connected component of the dependency graph, either all variables have value < 1 in the LFP or all variables have value $= 1$.

To classify the variables in the remaining system (after removing the variables of Z_0), we decompose the dependency graph into strongly connected components, and process the SCC's bottom-up, one at a time. Consider an SCC C and the set of equations $\{x_i = P_i(x)\}$ for the variables $x_i \in C$.

First, suppose that a variable x_i of type L has $P_i(x) = a_{i,0} + \sum_{j=1}^{n} a_{i,j} x_j$ where the sum of the coefficients $\sum_{j=0}^{n} a_{i,j} < 1$; we call such a variable *deficient*. Clearly then $P_i(x) < 1$ for any $x \in [0,1]^n$, hence $q_i^* = P_i(q^*) < 1$. Thus, if C contains a deficient variable, then all the variables of C have value < 1 and can be assigned to Z_b.

Second, suppose that for some $x_i \in C$, the function $P_i(x)$ contains a variable x_j from a lower SCC that is in Z_b, i.e., $q_j^* < 1$. Then $q_i^* < 1$ by Corollary A.2, and hence all the variables of C have value < 1 and can be assigned to Z_b.

Thus, we may assume that C does not contain any deficient variable and that all variables from lower SCCs that appear in $P_i(x)$ for $x_i \in C$ have value 1 in the LFP (belong to Z_1). Substitute 1 in their place in the functions $P_i(x)$. Since C does not contain any deficient variables, we have

$P_i(1) = 1$ for all $x_i \in C$. Also $q_i^* > 0$ for all $x_i \in C$ since we have eliminated the variables with value 0 in the LFP.

Thus, it remains to show how to classify a single (bottom) SCC C, whose equations $x = P(x)$ have LFP $q^* > 0$ and which satisfy $P(1) = 1$.

In the algorithm and the analysis, it will be convenient to use another function $Q : \mathbb{R}^n \to \mathbb{R}^n$ derived from P. We will use this function also in the next section for the case of maxPPSs, so we define it in general for all max-minPPSs (not necessarily strongly connected).

Definition A.3. *For a max-minPPS $x = P(x)$, define the function $Q : \mathbb{R}^n \to \mathbb{R}^n$ as follows. For each variable x_i, we define $Q_i(x)$ according to the type of x_i:*

- *Type L: If $P_i(x) = a_{i,0} + \sum_{j=1}^n a_{i,j}x_j$ then $Q_i(x) = \sum_{j=1}^n a_{i,j}x_j$.*
- *Type Q: If $P_i(x) = x_j x_k$, then $Q_i(x) = x_j + x_k$.*
- *Type Min: If $P_i(x) = \min(x_j, x_k)$ then $Q_i(x) = \max(x_j, x_k)$.*
- *Type Max: If $P_i(x) = \max(x_j, x_k)$ then $Q_i(x) = \min(x_j, x_k)$.*

Example A.1 Consider the minPPS of Example 3.1:

$$P(x) = (0.2x_2 + 0.3x_3 + 0.5,\ 0.4x_1 + 0.1x_3 + 0.5x_4,\ \min(x_2, x_5),\ x_1x_3,\ x_1^2).$$

Then,

$$Q(x) = (0.2x_2 + 0.3x_3,\ 0.4x_1 + 0.1x_3 + 0.5x_4,\ \max(x_2, x_5),\ x_1 + x_3,\ 2x_1).$$

□

Note that the function $Q(x)$ is homogeneous: for any scalar $\alpha > 0$ and any $x \geq 0$, we have $Q(\alpha x) = \alpha Q(x)$. For a minPPS, $Q(x)$ is a mixture of max expressions and linear expressions with no constant term and non-negative coefficients. Similarly, for a maxPPS, $Q(x)$ is a mixture of min expressions and linear expressions with no constant term. If $x = P(x)$ is a PPS, then we have $Q(x) = P'(1)x$, where $P'(1)$ is the moment matrix of P.

We will use the following simple lemma several times.

Lemma A.4. *For a minPPS $x = P(x)$, if the vector $v \geq 0$ has $v_j > 0$ and x_j appears in $P_i(x)$ then $Q_i(v) > 0$.*

Proof. If x_i is of type Min, i.e., $P_i(x) = \min(x_j, x_k)$ for some x_k, then $Q_i(v) = \max(v_j, v_k) \geq v_j > 0$. The cases of type-L and type-Q variables x_i are also immediate from the definition of Q. □

To classify an SCC with LFP > 0 and no deficient variables, we use the following algorithm:

Algorithm Qual-SCC
Input: A strongly-connected minPPS $x = P(x)$ with $q^* > 0$ and $P(1) = 1$.
Output: Decision whether $q^* = 1$ or $q^* < 1$.
Find an optimal solution to the linear program: maximize $\sum_i v_i$ subject to $Q(v) \leq v$ and $0 \leq v \leq 1$ for $v \in \mathbb{R}^n$.
If the optimal solution v is $v > 0$ then output '$q^* = 1$'.
If the optimal solution $v = 0$, then output '$q^* < 1$'.

For example, if we apply the algorithm to the minPPS of Example A.1 (it is strongly connected), we see that the optimal solution of the LP is 0, hence $q^* < 1$.

First we note that the above optimization problem is indeed a linear program: for a type L or type Q variable x_i, the function $Q_i(v)$ is linear, and for a type Min variable, the constraint $Q_i(v) = \max(v_j, v_k) \leq v_i$ is equivalent to the linear constraints $v_j \leq v_i$ and $v_k \leq v_i$.

Second, we note that the LP is feasible: The point 0 is a feasible solution since $P(0) = 0$. The objective function is bounded from above by n, hence the LP has an optimal solution.

Next we show that for every feasible solution v, either $v > 0$ or $v = 0$. To see this, suppose that $v \neq 0$, and let v_j be any coordinate with $v_j > 0$. If x_j appears in $P_i(x)$, then, from Lemma A.4 we have $v_i \geq Q_i(v) > 0$, i.e. if x_i has an edge to x_j in the dependency graph then also $v_i > 0$. Hence, by induction, the same is true for any variable x_i that can reach x_j. Since the system is strongly connected, this means that all $v_i > 0$, i.e. $v > 0$. In particular any optimal solution v to the LP satisfies either $v > 0$ or $v = 0$, so the algorithm always gives an output.

It remains to show that the algorithm's output is correct. The following lemma gives the key properties of Q. It shows that the operator Q plays for minPPSs (and branching Min MDPs) a role analogous to that played by the moment matrix $P'(1)$ for PPSs (and branching processes). Note that if $x = P(x)$ is a strongly-connected PPS, then the lemma reduces to the well-known spectral radius test on $P'(1)$.

Lemma A.5. *For a strongly connected minPPS $x = P(x)$ with $q^* > 0$ and $P(1) = 1$, the function $Q(v)$ satisfies:*

(i) There exists $\lambda > 0$ and $v > 0$ with $Q(v) = \lambda v$

(ii) $q^* = 1$ if and only if there is a non-zero $v \geq 0$ with $Q(v) \leq v$.

(iii) $q^* < 1$ if and only if there is a $v \geq 0$ with $Q(v) > v$.

Proof. Part (i). Firstly, we show that if $v \geq 0$ but $v \neq 0$, then $Q(v) \geq 0$ but $Q(v) \neq 0$. From the definition of Q, clearly $v \geq 0$ implies $Q(v) \geq 0$. Furthermore, if $v_j > 0$, by strong connectivity, there is an x_i such that x_j appears in $P_i(x)$, so $Q_i(v) > 0$ by Lemma A.4.

Consider the function $F(v) = \frac{Q(v)}{\|Q(v)\|_1}$ on the points of the unit simplex $\Delta = \{v | v \geq 0, \|v\|_1 = 1\}$. For all points $v \in \Delta$, we have that $v \geq 0$ and $v \neq 0$, therefore $Q(v) \geq 0$ and $Q(v) \neq 0$; hence $\|Q(v)\|_1 > 0$ and the function F is well-defined. Furthermore, F is clearly a continuous function and $\|F(v)\|_1 = 1$, thus F maps Δ to itself. So by Brouwer's fixed point theorem, there is a $v \in \Delta$ with $v = \frac{Q(v)}{\|Q(v)\|_1}$. The fixed point v satisfies $Q(v) = \lambda v$ for $\lambda = \|Q(v)\|_1 > 0$. Clearly $v \geq 0$ and $v \neq 0$. If $v_j > 0$ and x_j appears in $P_i(x)$ then, by Lemma A.4, $Q_i(v) > 0$, hence $v_i = \frac{Q_i(v)}{\|Q(v)\|_1} > 0$. Since the system is strongly connected, it follows by transitivity that all coordinates of v are positive, i.e. $v > 0$. This establishes part *(i)*.

We will next show the (if) directions of parts *(ii)* and *(iii)*, and then we will use them to deduce the (only if) directions.

Part (ii), if direction. Suppose that there is a non-zero $v \geq 0$ with $Q(v) \leq v$. We assume without loss of generality (by scaling) that $v \leq 1$. Let v_j be a positive coordinate of v. If x_j appears in $P_i(x)$, then by Lemma A.4, $Q_i(v) > 0$, hence also $v_i \geq Q_i(v) > 0$. Since the system is strongly connected, it follows again by transitivity that $v > 0$.

Consider any policy σ. Let $x = P_\sigma(x)$ be the induced PPS, and Q_σ the corresponding function. For every variable x_i of type L or Q, we have $(Q_\sigma)_i(v) = Q_i(v)$, and for every variable x_i of type M with function $P_i(x) = \min(x_j, x_k)$, we have $Q_i(v) = \max(v_j, v_k) \geq v_{\sigma(i)} = (Q_\sigma)_i(v)$. Consequently,

$Q(v) \geq Q_\sigma(v) = P'_\sigma(1)v$. Since $Q(v) \leq v$, we have that $P'_\sigma(1)v \leq v$. Since $v > 0$, by Perron-Frobenius theory (see Theorem 2.1.11 of [3]), it follows that $\rho(P'_\sigma(1)) \leq 1$.

The assumption that $q^* > 0$ implies in particular that the LFP of the PPS $x = P_\sigma(x)$ for the policy σ also satisfies $q^*_\sigma > 0$. We know that if the LFP of a PPS is > 0, then a variable x_i has value 1 in the LFP iff the moment matrix of the subsystem induced by the variables reachable from x_i has spectral radius ≤ 1 [18]. This is true for all the variables of the system $x = P_\sigma(x)$, since $q^*_\sigma > 0$ and the spectral radius of the whole moment matrix $\rho(P'_\sigma(1)) \leq 1$. Therefore, $q^*_\sigma = 1$. Thus we have shown that $q^*_\sigma = 1$ for any policy σ. This holds in particular for an optimal policy σ, hence $q^* = q^*_\sigma = 1$.

Part (iii), if direction. Suppose that there is a $v \geq 0$ with $Q(v) > v$. Note that then v has to be nonzero because $Q(0) = 0$. Let σ be a policy with $Q_\sigma(v) = Q(v)$; that is, for each type-M variable x_i with corresponding function $P_i(x) = \min(x_j, x_k)$, thus $Q_i(v) = \max(v_j, v_k)$, the policy σ sets $\sigma(i) = j$ if $v_j \geq v_k$, and $\sigma(i) = k$ otherwise. Then, $Q(v) = Q_\sigma(v) = P'_\sigma(1)v$. Since $Q(v) > v$, it follows that $P'_\sigma(1)v > v$. Noting again that $v \geq 0$ and $v \neq 0$, by Perron-Frobenius theory (see again Theorem 2.1.11 of [3]), this implies that $\rho(P'_\sigma(1)) > 1$. There is thus a principal irreducible submatrix $(P'_\sigma(1))_S$ of $P'_\sigma(1)$ with $\rho((P'_\sigma(1))_S) = \rho(P'_\sigma(1)) > 1$. Then S is an SCC of $x = P_\sigma(x)$ which has $(q^*_\sigma)_S < 1$. Thus $q^*_S \leq (q^*_\sigma)_S < 1$. Since $x = P(x)$ is a strongly connected minPPS, we have that $q^* < 1$.

Part (ii), only if direction. Suppose that $q^* = 1$. By part *(i)* we know that there exists a $\lambda > 0$ and a $v > 0$ such that $Q(v) = \lambda v$. If $\lambda > 1$, then $Q(v) > v$, which by the (if) direction of part *(iii)* implies that $q^* < 1$, a contradiction. Therefore, $\lambda \leq 1$. Hence, v satisfies $Q(v) \leq v$.

Part (iii), only if direction. The argument is similar. Suppose that $q^* < 1$. By part *(i)* we know that there exists a $\lambda > 0$ and a $v > 0$ such that $Q(v) = \lambda v$. If $\lambda \leq 1$, then $Q(v) \leq v$, which by the (if) direction of part *(ii)* implies that $q^* = 1$, a contradiction. Therefore, $\lambda > 1$. Hence, v satisfies $Q(v) > v$. □

We can show now the correctness of Algorithm Qual-SCC.

Lemma A.6. *Algorithm Qual-SCC classifies correctly a strongly connected minPPS $x = P(x)$ that has LFP $q^* > 0$ and satisfies $P(1) = 1$.*

Proof. We argued already that the algorithm always produces an output, either '$q^* = 1$' or '$q^* < 1$'. It remains to show that it produces the correct output, i.e., it outputs '$q^* = 1$' if and only if indeed the LFP $q^* = 1$.

(only if). Suppose that the algorithm outputs '$q^* = 1$', i.e. finds an optimal solution $v > 0$. Since $Q(v) \leq v$, Lemma A.5 (ii) implies $q^* = 1$.

(if). Suppose that $q^* = 1$. Then by part (ii) of Lemma A.5, there is a non-zero $v \geq 0$ with $Q(v) \leq v$. Since for a scalar $\alpha > 0$ and any $v \geq 0$, we have $Q(\alpha v) = \alpha Q(v)$, we may assume wlog that $\|v\|_\infty = 1$. Then $v \leq 1$. Since also $Q(v) \leq v$, the point v is a feasible solution of the LP. Since $v \neq 0$, the value of the objective function for v is greater than 0 and hence 0 is not an optimal solution to the LP. Since any feasible solution to the LP satisfies either $v = 0$ or $v > 0$, any optimal solution has $v > 0$. So the algorithm outputs '$q^* = 1$'. □

Summarizing, the algorithm for a general minPPS is as follows.

Algorithm Qual-minPPS

1. Compute the set Z_0 of variables that have value 0 in the LFP, and remove them from the system.
2. Partition the dependency graph of the remaining system into strongly connected components

and process the SCCs bottom-up in topological order.
3. For each SCC C, do the following:
3a. If some variable of C is deficient or depends on a variable that has been assigned already to Z_b then add all the variables of C to Z_b
3b. Else, apply algorithm Qual-SCC to the subsystem for the variables of C, substituting 1 for all variables from lower SCCs that appear on the right-hand side. If the LFP is 1, then add all the variables of C to Z_1, else add them to Z_b.

Let n be the number of variables of the given minPPS $x = P(x)$, let m be the total number of monomial terms in $P(x)$, and let L be the total number of bits needed to write down the given system. Let $T_{LP}(n, m, L)$ be the time needed to solve a Linear Program with n variables, m constraints and total bit-size L. For example, the Ellipsoid algorithm runs in $O(n^4L)$ time, and Karmakar's algorithm runs in $O(n^{3.5}L)$ time. Note that it is in general better to solve several small individual LPs rather than one large LP that combines the sizes of all the small LPs. Formally, we assume that T_{LP} is superadditive, i.e. $T_{LP}(n_1, m_1, L_1) + T_{LP}(n_2, m_2, L_2) \leq T_{LP}(n_1 + n_2, m_1 + m_2, L_1 + L_2)$; this is certainly the case with all the existing algorithms (they run in superlinear time).

Theorem A.7. *Algorithm Qual-minPPS classifies correctly all the variables of a minPPS (in SNF) and runs in polynomial time, specifically in time at most $T_{LP}(O(n), O(n), O(L))$, where n is the number of variables and L the bit-size of the minPPS.*

Proof. Correctness follows from our previous analysis. As for the running time, steps 1, 2 take time $O(n + m)$. Step 3a over all SCCs also takes time $O(n + m)$. For some SCCs C, the algorithm may execute step 3b and solve an LP with $O(n_C)$ variables and constraints and total bit-size $O(L_C)$, where n_C is the number of variables in C, and L_C is the bit-size of the subsystem for C. Since $\sum_C n_C \leq n$ and $\sum_C L_C \leq L$, by superadditivity of T_{LP}, the total time spent in step 3b over all SCCs is $T_{LP}(O(n), O(n), O(L))$. This dominates clearly the $O(n+m)$ time of the other steps (since $L \geq n + m$). Thus, the total time is $T_{LP}(O(n), O(n), O(L))$. □

It is worth mentioning that the dual of the LPs we have used in the Qual-minPPS algorithm are closely related (but not identical) to the linear system of inequalities that were used in the algorithm of [19] for deciding $q_i^* = 1$ for min-PPSs. Furthermore, the dual LP can be used in order to compute a (deterministic) policy σ for the minimizing player which forces $(q_\sigma^*)_i < 1$ whenever $q_i^* < 1$.

An alternative approach for computing such a policy σ is to use repeated calls to the Qual-minPPS algorithm, as follows. First, call Qual-minPPS on the entire minPPS, $x = P(x)$. Let Z_b be the output of the algorithms. Pick some $x_i \in Z_b$ which has type-M (if none exists, then the algorithm is finished). Suppose $P_i(x) \equiv \min(x_j, x_k)$. Tentatively fix the "action" at x_i to be $\to x_j$, by setting $P_i(x) := x_j$. Re-run Qual-minPPS on this revised min-PPS, computing a new set Z_b. If x_i remains in Z_b, then fix the action choice x_j permanently, i.e., leave $P_i(x) := x_j$. Otherwise, fix action $\to x_k$ instead, i.e., let $P_i(x) := x_j$. Do this repeatedly, until there are no remaining type-M variables in Z_b. The algorithm terminates after at most n calls to Qual-minPPS. It can be shown that the action choices produced by this algorithm define a deterministic policy σ (in which we can let the action choice at type-M nodes in Z_1 be arbitrary) such that $(q_\sigma^*)_i < 1$ if and only if $q_i^* < 1$, for all variables x_i.

General form minPPSs.

If the given minPPS $x = P(x)$ is in general form, we could transform it to SNF and apply the above algorithm. The transformation changes the encoding size of the system linearly, however it can increase substantially the number of variables. For this reason, it is better to deal directly with the given system. The algorithm is essentially the same as the one for the SNF case. The only difference is that we have to define the operator Q for general systems, and define the term 'deficient' in this context.

Given a max-minPPS $x = P(x)$ in general form, we define the operator Q as follows: If $P_i(x) = \min\{p_{ij}(x)|j \in \{1,\ldots,m_i\}\}$, where $p_{ij}(x) = a_{ij,0}+\sum_{r\in R(i,j)} a_{ij,r}x^{\alpha_{ij,r}}$, then $Q_i(v) = \max\{\hat{p}_{ij}(v)|j \in \{1,\ldots,m_i\}\}$ where $\hat{p}_{ij}(v) = \sum_{r\in R(i,j)} a_{ij,r}\alpha_{ij,r} \cdot v$. If $P_i(x) = \max\{p_{ij}(x)|j \in \{1,\ldots,m_i\}\}$, then $Q_i(v) = \min\{\hat{p}_{ij}(v)|j \in \{1,\ldots,m_i\}\}$. Note that the polynomials $\hat{p}_{ij}(v)$ are homogeneous (the constant term is 0). For example, if $p_{ij}(x) = 0.2 + 0.3x_1^2x_2^3 + 0.5x_1^3x_2x_3^4$, then $\hat{p}_{ij}(v) = 2.1v_1 + 1.4v_2 + 2v_3$. If $x = P(x)$ is a PPS (in general form) then $Q(v) = P'(1) \cdot v$, where $P'(1)$ is the moment matrix of P.

Call a polynomial $p_{ij}(x)$ *deficient* if the sum of its coefficients is < 1; a variable x_i of type Min is deficient if some polynomial in $P_i(x)$ is deficient. (For variables of type Max, x_i is deficient if all the polynomials in $P_i(x)$ are deficient.)

Given these definitions, we apply the same Algorithm Qual-minPPS to a general form minPPS to classify its variables into Z_0, Z_b and Z_1. The proof of correctness is exactly the same as in the SNF case. Steps 1, 2, and 3a take time $O(L)$ where $L = |P|$ is the encoding size of the given system. In Step 3b, we solve LPs of the form maximize $\sum_i v_i$ subject to $Q(v) \le v$ and $0 \le v \le 1$ for some SCCs C of the dependency graph. The LP for an SCC C has one variable for every variable x_i of C, and has constraints $\hat{p}_{ij}(v) \le v_i$ for each $j = 1,\ldots,m_i$. Thus, the number of variables of the LP for C is $n_C = |C|$, the number of constraints is $m_C = \sum_{x_i\in C} m_i$, and its size is $O(L_C)$, linear in the encoding size of the subsystem for C. By the superadditivity of the time T_{LP} needed to solve LP's we have:

Theorem A.8. *We can classify qualitatively the variables of a minPPS $x = P(x)$ (in general form) in time $T_{LP}(O(n), O(m), O(L))$, where n is the number of variables, m is the number of polynomials and L is the encoding size of the minPPS.*

A.2 Qualitative classification for maxPPSs

We are given a maxPPS $x = P(x)$, without loss of generality, in simple normal form. (We will discuss again at the end of the subsection the case of maxPPS in general form.) Let q^* be its LFP. We want to partition the variables into the sets $Z_0 = \{x_i \mid q_i^* = 0\}$, $Z_b = \{x_i \mid 0 < q_i^* < 1\}$, and $Z_1 = \{x_i \mid q_i^* = 1\}$. Let $L_{<1}$ be the set of *deficient* variables, i.e., the set of type-L variables x_i whose function $P_i(x) = a_{i,0} + \sum_{j=1}^n a_{i,j}x_j$ has $P_i(1) = \sum_{j=0}^n a_{i,j} < 1$. Recall, a type-L variable x_i is called *leaky* if $P_i(0) = a_{i,0} > 0$ and we let $L_{>0}$ be the set of all leaky variables. Let G be the dependency graph of the system $x = P(x)$.

The algorithm has two phases. In the first phase we identify and remove the set Z_0 and a subset of Z_b. The remaining system on the rest of the variables is not quantitatively equivalent to the original system, i.e. it does not have the same LFP on the remaining variables, but it is qualitatively equivalent, i.e. it has the same partition into Z_b and Z_1. The reduced system does not contain any deficient variables and has LFP > 0. Phase 2, which is the heart of the algorithm, processes the remaining system and partitions its variables into Z_b and Z_1.

Phase 1 Algorithm

Input: A maxPPS $x = P(x)$
Output: Assignment of a subset of variables to Z_0 and Z_b and a reduced maxPPS on the remaining variables

1. Compute the set Z_0 of variables that have value 0 in the LFP and remove them from the system.

2. Compute the set of variables that are deficient or that can reach the set $L_{<1}$ of deficient variables by and-or reachability on G, where type L and type Q variables are "or" nodes and type M variables are "and" nodes. Add the variables thus found to Z_b and remove them from the system (both from the left- and right-hand sides).

3. Compute the set of variables that can reach in G the set $L_{>0}$ of leaky variables by and-or reachability where type L and type M variables are "or" nodes and type Q variables are "and" nodes. Add all other variables (i.e., those that *cannot* reach $L_{>0}$) to Z_b and remove them from the system (from both sides).

4. If there was a change in step 3, go back to step 2, else terminate this phase.

In step 1 we identify and remove the set Z_0 of variables that have value 0 in the LFP. The remaining variables belong to Z_b or Z_1. If a variable x_i is deficient, then clearly $q_i^* = P_i(q^*) \leq P_i(1) < 1$ and therefore x_i belongs to Z_b. In the case of maxPPS, Corollary A.2 does not hold for ordinary graph reachability, but the analogous property holds for and-or reachability as in step 2.

Lemma A.9. *1. If x_i is a type L or type Q variable and $P_i(x)$ contains a variable x_j with $q_j^* < 1$ then $q_i^* < 1$. The same is true if x_i is of type M and all variables x_j in $P_i(x)$ have $q_j^* < 1$.*
2. If $S \subseteq Z_b \cup Z_0$ and x_i can reach S via and-or reachability where type L and type Q variables are "or" nodes and type M variables are "and" nodes, then also $x_i \in Z_b \cup Z_0$.

Proof. It is straightforward to verify part 1. Part 2 follows by induction. □

Thus, step 2 of the Phase 1 algorithm correctly assigns the variables computed in the and-or reachability to Z_b. When we remove these variables from the right-hand sides of the equations for the remaining variables, we get a new system which in general does *not* have the same fixed point. Note that if a variable x_j is removed in step 2 and x_j appeared in $P_i(x)$, if x_i is of type L or Q, then also x_i is removed, but if x_i is of type M with corresponding function $P_i(x) = \max(x_j, x_k)$ and x_k is not removed, then x_i is not removed either and its function is changed to $P_i(x) = x_k$. This in effect forces the policy to select x_k for the variable x_i which may not be the optimal choice, and as a consequence, the LFP of the reduced system may give a lower value to some variables than the LFP of the original system. However, the variables that had value 1 will keep their value:

Lemma A.10. *Let $x = P(x)$ be a maxPPS and $S \subseteq Z_b \cup Z_0$. If we remove the variables in S from the system setting them to 0, the reduced system has the same set of variables with value 1 in the LFP as the original system.*

Proof. Let σ be an optimal policy for the original system. Then the LFP q_σ^* of the system $x = P_\sigma(x)$ satisfies $q_\sigma^* = q^*$. Consider the dependency graph G_σ of $x = P_\sigma(x)$. By Corollary A.2, the variables of Z_1 can reach in G_σ only variables of Z_1. Hence, σ, restricted to Z_1, is a valid policy in the reduced system for the variables of Z_1, which will assign value 1 to all these variables. □

After removing the variables computed in step 2, the LFP of the reduced system may give value 0 to some variables. Consider for example a system with equations $x_1 = \max(x_2, x_3)$, $x_2 = \max(x_1, x_3)$, $x_3 = 0.3 + 0.5x_1$. Variable x_3 is deficient, thus Step 2 will remove it and reduce the system to $x_1 = x_2$, $x_2 = x_1$, whose LFP gives value 0 to both variables. Step 3 computes the set of variables that have value 0 in the LFP of the new system. It assigns them to Z_b correctly because we know that their value in the LFP of the original system is not 0 (otherwise they would have been assigned in Step 1) and is not 1 (otherwise it would be also 1 in the current system by Lemma A.10). Step 3 removes them from the system, which does not affect the set Z_1 by Lemma A.10. Their removal may create new deficient variables. For example, consider the above system along with an additional equation $x_4 = 0.5x_1 + 0.5$. After removing x_1, the variable x_4 becomes deficient. Thus, if some variables are removed in step 3, the algorithm returns to step 2 to test if there are more deficient variables to remove them along with possibly more variables.

When the Phase 1 algorithm terminates, the reduced system has the properties given in the following lemma. Let n be the number of variables, m the total number of terms in $P(x)$ and L the bit-size of the system.

Lemma A.11. *1. The Phase 1 algorithm runs in polynomial time, specifically in time $O(n(n+m))$.*
2. Every variable x_i assigned by it to Z_0 has value $q_i^* = 0$ in the LFP and every variable assigned to Z_b has value $0 < q_i^* < 1$.
3. The reduced system $x = \hat{P}(x)$ at the end of the Phase 1 algorithm has LFP $\hat{q}^* > 0$ and satisfies $\hat{P}(1) = 1$. Furthermore, $Z_1 = \{x_i | q_i^* = 1\} = \{x_i | \hat{q}_i^* = 1\}$.

Proof. 1. Step 1 takes linear time, $O(n + m)$. Also every execution of Step 2 or 3 takes linear time, and all of the executions, except for the last one, remove some variables, hence there are $O(n)$ executions. Thus, the total time is at most $O(n(n + m))$.

2. Step 1 computes all the variables of Z_0, and as we argued earlier, all the variables identified and removed in Steps 2 and 3 belong to Z_b.

3. After Step 2, the current system has no deficient variables and thus satisfies $P(1) = 1$, and after Step 3, the LFP of the current system is > 0. Thus, if in an iteration, Step 3 does not remove any more variables and the algorithm terminates, then the system satisfies both properties. The set of variables that have value 1 in the LFP does not change in any step by Lemma A.10, therefore this set in the final system is the same as in the original system. □

We proceed now to Phase 2. The algorithm in this phase repeatedly computes subsets of variables that have value 1 in the LFP and reduces the system, substituting 1 for their value, until it cannot find any more such variables, at which point the remaining variables are assigned to Z_b, and the algorithm terminates. Recall the function Q defined from the function P in Definition A.3.

Phase 2 Algorithm
Input: A maxPPS $x = P(x)$ in SNF with $P(1) = 1$ and $q^* > 0$.
Output: Partition of the variables into Z_1 and Z_b according as their value in the LFP q^* is $= 1$ or < 1.

Let X be the set of all variables.

1. Find an optimal solution of the LP: maximize $\sum_i v_i$ subject to $Q(v) \geq v$ and $0 \leq v \leq 1$. Call this optimal solution v.

2. Add all variables x_i with $v_i = 0$ to Z_1. Eliminate all variables added to Z_1 from the maxPPS by substituting 1.

3. Set $Y = X \setminus Z_1$. Then eliminate from Y all variables x_i that have $Q_i(v) > v_i$.

4. Eliminate from Y all variables x_i that can reach in G the set $X \setminus (Z_1 \cup Y)$ via and-or reachability with type `M` being "and" and types `Q` and `L` being "or".

5. Eliminate from `Y` all variables x_i which are not leaky and cannot reach the set $L_{>0}$ of leaky variables via and-or reachability in the subgraph $G[Y]$ of G induced by Y, with type Q being "and" and types L and M being "or", and where for each variable x_i of type `M` we can only use choices $x_j \in Y$ with $v_j = v_i$ for this reachability.

6. If any variables were removed from Y in step 5, return to step 4

7. Add all variables that remain in Y to Z_1. If any variables were added to Z_1, eliminate all such variables from the maxPPS by substituting 1.

8. If any variables were added to Z_1 in steps 2 or 7, return to 1.

9. Add all remaining variables to Z_b.

Note that the optimization problem in Step 1 is indeed a Linear Program: the function $Q(v)$ for a type L or a type Q variable is linear; for a type M variable x_i, the constraint $Q_i(v) \geq v_i$ has the form $\min(v_j, v_k) \geq v_i$, which is equivalent to the conjunction of the constraints $v_j \geq v_i$ and $v_k \geq v_i$. Furthermore, the LP is feasible: the point 0 is a feasible solution since $Q(0) = 0$. Since the objective function is bounded from above by n, there is an optimal solution.

Lemma A.12. *The Phase 2 algorithm runs in polynomial time. Specifically, for a system with n variables, m terms on the right-hand side and bit-size L, the time is bounded by $O(n^2(n+m) + nT_{LP}(O(n), O(n), O(L)))$.*

Proof. The algorithm consists of the outer loop of steps 1-8 and the nested inner loop of steps 4-6. Every execution of the outer loop, except for the last one, adds some more variables to Z_1, hence there are at most $n+1$ executions. In every execution of the outer loop, the inner loop starts with an initial set Y and keeps removing variables from Y in every iteration, except for the last one, hence there are at most $n+1$ iterations of the inner loop in every execution of the outer loop. Thus, there is a total of at most $(n+1)^2$ executions of the inner loop and of each step throughout the algorithm.

Step 1 involves solving an LP with $O(n)$ variables and constraints and of bit-size linear in the size L of the given maxPPS, and takes time $T_{LP}(O(n), O(n), O(L))$. This step is executed at most $n+1$ times. Each and-or reachability step takes time $O(n+m)$, and is executed at most $(n+1)^2$ times. Therefore, the algorithm runs in polynomial time, specifically in the time indicated in the lemma. □

The following two lemmas show now the correctness of the Phase 2 algorithm, i.e. that all variables x_i added to Z_1 have indeed value $q_i^* = 1$, and conversely, all variables with value $q_i^* = 1$ are eventually added to Z_1.

Lemma A.13. *Every variable x_i that is added by the Phase 2 algorithm to Z_1 has value $q_i^* = 1$ in the LFP.*

Proof. We use induction on the time of the addition to Z_1. Suppose that all variables that have been added so far to Z_1 have indeed value 1 in the LFP. Then the current reduced system on the remaining variables is (quantitatively) equivalent to the given input system of the Phase 2 algorithm. To simplify notation, we will still use $x = P(x)$ for the current system and q^* for its LFP. Note that substituting 1 for some variables in Z_1 does not affect the properties $q^* > 0$ and $P(1) = 1$. We consider separately the variables added to Z_1 in steps 2 and 7.

Variables added to Z_1 in step 2.

We claim that $1 - q^*$ is a feasible solution of the LP in step 1. For a variable x_i of type `L` we have, $Q_i(1 - q^*) = \sum_{j=1}^{n} a_{i,j}(1 - q_j^*) = \sum_{j=1}^{n} a_{i,j} - \sum_{j=1}^{n} a_{i,j}q_j^* = 1 - a_{i,0} - \sum_{j=1}^{n} a_{i,j}q_j^* = 1 - P_i(q^*) = 1 - q_i^*$, where we have used the property that $P(1) = 1$. For a variable x_i of type `M` we have $P_i(x) = \max(x_j, x_k)$ for some x_j, x_k, and $Q_i(1 - q^*) = \min(1 - q_j^*, 1 - q_k^*) = 1 - \max(q_j^*, q_k^*) = 1 - P_i(q^*) = 1 - q_i^*$. For a variable x_i of type `Q`, we have $P_i(x) = x_j x_k$ for some x_j, x_k and $Q_i(1 - q^*) = (1 - q_j^*) + (1 - q_k^*) = 1 + (1 - q_j^*)(1 - q_k^*) - q_j^* q_k^* \geq 1 - q_j^* q_k^* = 1 - q_i^*$. Hence, $Q(1 - q^*) \geq 1 - q^*$, and so $1 - q^*$ is a feasible solution to the LP.

Next we claim that the LP has a unique optimal solution v and $v \geq 1 - q^*$. Note that $Q(v)$ is monotone, i.e. if $0 \leq v_1 \leq v_2 \leq 1$ then $Q(v_1) \leq Q(v_2)$. For any two feasible solutions of the LP v_1,v_2, let v_3 be the coordinate-wise maximum of v_1,v_2. Then $Q(v_3) \geq Q(v_1) \geq v_1$. Similarly $Q(v_3) \geq v_2$ and so $Q(v_3) \geq v_3$. It follows that the set of feasible solutions has a coordinate-wise maximum that is the unique optimal solution v. Since $1 - q^*$ is feasible, we have that the optimal solution v satisfies $v \geq 1 - q^*$. Thus if $v_i = 0$ then $q_i^* = 1$. So all variables x_i added to Z_1 in step 2 have $q_i^* = 1$.

Variables added to Z_1 in step 7.

Consider the final iteration of the inner loop of steps 4-6 before proceeding to step 7. After step 3, all variables x_i in Y have $Q_i(v) = v_i$. After step 4, if a variable x_i with type `L` or `Q` is in Y then all variables in $P_i(x)$ are also in Y, and if x_i is of type `M` then at least one variable of $P_i(x)$ is also in Y. The same is true when we complete the final iteration of the inner loop and continue past step 6 because then the set Y has not been modified in steps 5 and 6.

After step 5, every variable in Y can reach $L_{>0}$ via and-or reachability in $G[Y]$, with type Q being "and" and types L and M being "or", and using for type-M variables x_i only a successor $x_j \in P_i(x)$ with $v_i = v_j$. Thus, for each variable $x_t \in Y$ there is a reachability 'proof' in the form of a DAG (directed acyclic graph) $D(x_t)$ that is a subgraph of $G[Y]$, and which has x_t as the (unique) source node, all the sinks are leaky nodes, every type-Q node $x_i \in D(x_t)$ has both its outgoing edges present in the DAG, every internal type-L or type-M node $x_i \in D(x_t)$ has one outgoing edge $x_i \to x_j$ in $D(t)$ and if x_i is of type M then $v_i = v_j$. Measure the 'height' of the proof by the height of the DAG. Fix a shortest (minimum height) such reachability 'proof' for each variable in Y, and let σ be a policy for the type-M variables of Y that selects for each type-M variable x_i its successor x_j in the shortest reachability proof for x_i. Note that $v_i = v_j$, and the shortest reachability proof for x_j does not involve x_i.

Consider the system $x = P_\sigma(x)$ with 1 substituted for the variables already in Z_1, and its subsystem $x_Y = (P_\sigma(x))_Y$ consisting of the equations for the variables in Y. For every $x_i \in Y$, all variables of $(P_\sigma(x))_i$ are in Y; note that the equation for a type-M variable $x_i \in Y$ has become $x_i = x_j$ in $x = P_\sigma(x)$, where $j = \sigma(i)$ and $v_i = v_j$. Thus the subsystem $x_Y = (P_\sigma(x))_Y$ is a PPS on the set of variables x_Y, which we denote by $x_Y = P_\sigma(x_Y)$. From step 5 and our choice of σ, all variables of Y can reach $L_{>0}$ via and-or reachability on the dependency graph of $x_Y = P_\sigma(x_Y)$, with variables of type `Q` being "and" and variables of type `L` being "or". It follows that $(q_\sigma^*)_Y > 0$.

For all variables $x_i \in Y$, we have $(Q_\sigma(v))_i = v_i$, thus $(Q_\sigma(v))_Y = v_Y$. Let $(P'_\sigma(1))_Y$ be the moment matrix of $x_Y = P_\sigma(x_Y)$. We have $(P'_\sigma(1))_Y v_Y = (Q_\sigma(v))_Y = v_Y$. That is, the positive vector v_Y is an eigenvector of the non-negative (but not necessarily irreducible) matrix $(P'_\sigma(1))_Y$ with eigenvalue 1. It follows from standard facts of Perron-Frobenius theory (see Corollary 2.1.12 of [3]) that $\rho((P'_\sigma(1))_Y) = 1$. Since the LFP $(q^*_\sigma)_Y > 0$, it follows from [18][6] that $(q^*_\sigma)_i = 1$ for all $x_i \in Y$. Therefore, all variables x_i added to Z_1 in step 7 have $q^*_i = 1$. □

We show the converse now.

Lemma A.14. *Every variable with value 1 in the LFP is eventually added to Z_1 by the Phase 2 algorithm. Furthermore, every variable x_j that is added to Z_b has value in the LFP strictly between 0 and 1.*

Proof. We need to show that if in some iteration of the outer loop there is still an $x_i \notin Z_1$ with $q^*_i = 1$, then some variable is added to Z_1 in either step 2 or step 7. If the optimal solution v to the LP has $v_k = 0$ for some k, then x_k is added to Z_1 in step 2. So assume henceforth that $v_k > 0$ for all k, and that $q^*_i = 1$.

We consider the current maxPPS obtained by eliminating all those variables already in Z_1 by substituting 1. Let σ be an optimal policy. We have $(q^*_\sigma)_i = 1$ and all variables x_j that x_i depends on in $x = P_\sigma(x)$ also have $(q^*_\sigma)_j = 1$. Thus there is some bottom SCC S of $x = P_\sigma(x)$ that has $(q^*_\sigma)_S = 1$. Then we must have (by [18], Section 8.1, Lemma 8.4) that $\rho((P'_\sigma(1))_S) \leq 1$.

From the definition of Q, we have that $Q_\sigma(v) = P'_\sigma(1)v \geq Q(v)$ (for any policy σ), and we know that $Q(v) \geq v$. Therefore, $P'_\sigma(1)v \geq v$. Restricted to S, we still have $(P'_\sigma(1))_S v_S \geq v_S$. Since $(P'_\sigma(1))_S$ is a non-negative and irreducible matrix and $v_S > 0$, by standard facts of Perron-Frobenius theory (again, see Theorem 2.1.11 of [3]), $\rho((P'_\sigma(1))_S) \geq 1$. Therefore, $\rho((P'_\sigma(1))_S) = 1$.

We know that $(P'_\sigma(1))_S v_S \geq v_S$, where $(P'_\sigma(1))_S$ is irreducible and $v_S > 0$. It follows by standard facts of Perron-Frobenius theory that we must have $(P'_\sigma(1))_S v_S = v_S$, because otherwise, if there was strict inequality for any coordinate then it would imply $\rho((P'_\sigma(1))_S) > 1$ (again, see the last part of Theorem 2.1.11 of [3]). Since $(Q_\sigma(v))_S = (P'_\sigma(1))_S v_S = v_S$, we still have $S \subseteq Y$ after step 3.

All successors of the type-L and type-Q variables of S are also in S and the same is true of the σ-successors of type M variables, since S is a bottom SCC of $x = P_\sigma(x)$. Therefore, $S \subseteq Y$ after step 4.

Consider step 5. Since $(q^*_\sigma)_S = 1 > 0$, for any x_i in S, either x_i is itself a leaky variable or x_i can reach the set $L_{>0}$ of leaky variables via and-or reachability in $G[Y]$ with type Q being "and" and type L being "or", and with every x_t of type `M` in S using the choice $x_j \in S$ where $j = \sigma(t)$; note that $v_t = (Q_\sigma(v))_t = v_j$. So we still have $S \subseteq Y$ after step 5 as well. The inner loop of steps 4-6 may be executed several times, but the set S will always remain in Y through all the iterations, as we have argued. Thus, once there are no more changes in step 5, the algorithm will exit the inner loop with $S \subseteq Y$ and all variables in S will be added to Z_1 in step 7.

Thus, we have shown that if at the beginning of an execution of the outer loop, there is an $x_i \notin Z_1$ with $q^*_i = 1$, then some variable is added to Z_1 in either step 2 or step 7. Since we only add variables to Z_1 and we only have a finite number of variables, eventually the algorithm will terminate. During the last execution of the outer loop, no more variables are added to Z_1, which implies that the set Z_1 at this point contains all the variables x_i with $q^*_i = 1$. Therefore, all

[6] See, in [18], the algorithm in Figure 7, and the last paragraph of Section 8.1 which establishes its correctness.

the remaining variables x_j that are added at the end to Z_b must have $q_j^* < 1$, and $q_j^* > 0$ since $q^* > 0$. □

Combining Lemmas A.11, A.12, A.13, A.14, we have:

Theorem A.15. *The algorithm consisting of phases 1 and 2 classifies correctly all the variables of a maxPPS (in SNF) in polynomial time, specifically in time* $O(n^2(n+m)+nT_{LP}(O(n),O(n),O(L)))$.

We will show next that we can compute within the same time bounds a (partial) policy σ under which all variables in Z_1 get value 1 in the LFP. For the variables that are added to Z_1 in step 7 of the Phase 2 Algorithm, we know that they can reach the set $L_{>0}$ of leaky variables in step 5. We compute their σ-successors in $O(n+m)$ time as we perform and-or reachability in step 5 to find the set R of variables that can reach the set $L_{>0}$ of leaky variables. The algorithm works in rounds. Initially, we set $R = L_{>0} \cap Y$. In each round we add to R all the variables x_i of $Y - R$ that are either (i) of type L and have an edge to some variable $x_j \in R$, or (ii) of type Q and both its edges go to variables in R, or (iii) of type M and at least one edge goes to a variable $x_j \in R$ with $v_j = v_i$; in the last case, set $\sigma(i) = j$ (if both edges of x_i go to such nodes in R then pick arbitrarily one of them). It is shown in the proof of Lemma A.13 that this policy ensures that all variables $x_i \in Y$ get value 1 in the LFP (provided that the variables assigned earlier to Z_1 get value 1).

It remains to show how to pick a policy for the variables added to Z_1 in step 2. Let DLP be the dual LP to the LP of step 1. The variables of DLP are as follows. For each variable x_i of the maxPPS, there is a variable z_i in the dual LP that is complementary to the constraint $v_i \leq 1$ of the primal; for each variable x_i of type L or type Q there is a dual variable y_i that is complementary to the constraint $Q_i(v) \geq v_i$; for each x_i of type M, where $P_i(x) = \max(x_j, x_k)$, there are two dual variables y_{ij}, y_{ik} complementary to the constraints $v_j \geq v_i$ and $v_k \geq v_i$.

Let V_L, V_Q, V_M be respectively the sets of type-L, type-Q, and type-M variables. For a variable x_i, we let $\Gamma^{-1}(x_i)$ be the set of variables that have an edge to x_i in the dependency graph. The dual LP is as follows.

minimize $\sum_i z_i$ subject to:

- For all $x_i \in V_L \cup V_Q$: $z_i + y_i \geq 1 + \sum\{a_{ti}y_t | x_t \in V_L \cap \Gamma^{-1}(x_i)\} + \sum\{y_t | x_t \in V_Q \cap \Gamma^{-1}(x_i)\} + \sum\{y_{ti} | x_t \in V_M \cap \Gamma^{-1}(x_i)\}$
- For all $x_i \in V_M$: $z_i + y_{ij} + y_{ik} \geq 1 + \sum\{a_{ti}y_t | x_t \in V_L \cap \Gamma^{-1}(x_i)\} + \sum\{y_t | x_t \in V_Q \cap \Gamma^{-1}(x_i)\} + \sum\{y_{ti} | x_t \in V_M \cap \Gamma^{-1}(x_i)\}$
- All variables $z_i, y_i, y_{ij} \geq 0$

Let v be the optimal solution of the primal LP of step 1, and let $S = \{x_i | v_i = 0\}$. From the primal constraints, if $x_i \in S$ is of type L or Q, then all its (immediate) successors x_j must have $v_j = 0$, i.e. $x_j \in S$, because otherwise we can raise the value of v_i and improve the objective function. If $x_i \in S$ is of type M, then at least one successor must be in S for the same reason.

Let LP_S be the primal LP restricted to the variables v_i and constraints for $x_i \in S$: If x_i is of type L or Q, or if it of type M and both successors are in S, then the corresponding constraint $Q(v)_i \geq v_i$ is the same as in the full primal LP (because all successors of x_i are in S). If x_i is of type M and only one successor x_j is in S then the corresponding constraint in LP_S is only $v_j \geq v_i$; that is, the other constraint $v_k \geq v_i$ of the full LP is not included in LP_S because the variable v_k is missing since $x_k \notin S$. Note that LP_S is the same as the primal LP for the maxPPS obtained by

restricting the given maxPPS $x = P(x)$ to the subset S, where the variables x_i of type M with only one successor x_j in S become type-L variables with equation $x_i = x_j$ (this corresponds essentially to fixing the policy for such a variable x_i to the successor x_j).

Let DLP_S be the dual LP to LP_S. This is the same as the LP obtained by restricting the full dual DLP to the constraints corresponding to the variables $x_i \in S$ and including only the variables z_i, y_i for $x_i \in S$ and the variables y_{ij} if both $x_i, x_j \in S$.

Lemma A.16. *The optimal value of the restricted linear programs LP_S, DLP_S is 0.*

Proof. Let v be the optimal solution for the full primal LP and consider any optimal solution (y, z) for the full dual DLP. By the complementary slackness conditions we have $z_i = 0$ for all $x_i \in S$, since $v_i = 0 < 1$.

We claim that the restriction (y_S, z_S) of (y, z) to the variables of DLP_S is a feasible solution for DLP_S. To see this, note that the constraints of DLP_S are a subset of the constraints of DLP, with some of the variables removed. The only removed variables that appear on the left-hand side of the inequalities are the variables y_{ik} where x_i is a type-M variable in S and x_k is not in S. Then $v_i = 0 < v_k$, and hence by complementary slackness we must have $y_{ik} = 0$. Thus, all the removed variables that appear on the left-hand sides of the inequalities have value 0. All the other removed variables appear on the right-hand sides of the inequalities with a positive coefficient, and hence after their removal, the inequalities continue to be satisfied by the restricted solution.

Thus, (y_S, z_S) is a feasible solution to DLP_S and its value is 0 since $z_i = 0$ for all $x_i \in S$. The restricted primal LP_S has obviously a feasible solution $v_S = 0$ with value 0, hence the optimal value for the two LPs is 0, and both v_S, (y_S, z_S) are optimal solutions for them. □

To compute an optimal policy for the type-M variables in S, we construct the restricted dual DLP_S and compute an optimal basic feasible solution. Let (y, z) be this solution. Since it is optimal, $z_i = 0$ for all i. From the dual constraints, for all type-L or type-Q variables x_i we have $z_i + y_i \geq 1$, hence at least one of the two variables z_i, y_i is positive, and since $z_i = 0$, we have $y_i > 0$. For all type-M variables x_i, we have $z_i + y_{ij} + y_{ik} \geq 1$, and since $z_i = 0$, at least one of the variables y_{ij}, y_{ik} is positive. Since (y, z) is a basic feasible solution, it has at most $|S|$ positive variables, hence it must have exactly one positive variable for each x_i. Therefore, for each $x_i \in V_M \cap S$, exactly one of y_{ij} or y_{ik} is positive; let σ be the policy that selects for each type-M variable of S the (unique) successor with a positive y value for the edge, i.e. x_j if $y_{ij} > 0$, else x_k. (If x_i is a type-M variable in S that has only one successor x_j in S, then obviously σ assigns to x_i this successor.)

Lemma A.17. *Under the policy σ, all the variables in S get value 1 in the LFP.*

Proof. Let $x_S = P_\sigma(x_S)$ be the PPS on the set S of variables induced by the policy σ, let G_σ be its dependency graph, $P'_\sigma(1)$ the moment matrix, and q^*_σ its LFP. Let $\hat{y}$ be the vector, indexed by S, that is derived from the optimal basic feasible solution (y, z) of DLP_S, by letting $\hat{y}_i = y_i$ for all type-L and type-Q variables x_i in S and letting $\hat{y}_i = y_{i,\sigma(i)}$ for all type-M variables in S. Then $\hat{y} > 0$. Furthermore, the constraints of DLP_S imply that $\hat{y} \geq 1 + [P'_\sigma(1)]^T \cdot \hat{y}$. Therefore, $\rho([P'_\sigma(1)]^T) = \rho(P'_\sigma(1)) < 1$ (again, see Theorem 2.1.11 of [3]). This implies that $q^*_\sigma = 1$. Let us see why. We first claim that $q^*_\sigma > 0$. If not, then the set W of variables that cannot reach in G_σ the set of leaky variables via and-or reachability, with type L as 'or' nodes and type Q as 'and' nodes, is a nonempty set. Every type-L variable of W is not leaky and has all its successors in W, and every type Q variable of W has at least one successor in W. This implies that in the submatrix $[P'_\sigma(1)]_W$

induced by the rows and columns of W, the sum of the entries in every row is at least 1. Hence $[P'_\sigma(1)]_W \cdot 1 \geq 1$, and thus $\rho([P'_\sigma(1)]_W) \geq 1$ (again, by Theorem 2.1.11 of [3]), contradicting the fact that $\rho(P'_\sigma(1)) < 1$. Therefore, $q^*_\sigma > 0$. Since we also have $\rho(P'_\sigma(1)) < 1$, it follows (see [18], Section 8.1), that $q^*_\sigma = 1$. □

The LP DLP_S has $O(|S|)$ variables and constraints and can be solved in time $T_{LP}(O(|S|), O(|S|), O(L_S))$ where L_S is its bit-size. Note that even if an LP solver outputs an optimal solution that is not a basic feasible solution (bfs), it is known that one can easily convert it to an optimal bfs using standard methods, by suitable pivoting steps that drive nonbasic variables to 0[7]. The time to solve all the dual programs DLP_S for all the executions of step 2 that add variables to Z_1 is at most $T_{LP}(O(n), O(n), O(L))$, by the superadditivity of T_{LP}. Thus, we have.

Theorem A.18. *We can compute an optimal policy for all the variables of a maxPPS (in SNF) that have value 1 in the LFP within the same time bounds as given in Theorem A.15 for the classification of the variables.*

General form maxPPS.

For maxPPS $x = P(x)$ in general form it is again more efficient to apply the algorithms directly to the given system, without transforming it first to SNF. The algorithms are essentially the same as in the SNF case. In Phase 1, we identify first the set Z_0 of variables that have value 0 in the LFP and eliminate them. Then we construct the refined graph G_r for the remaining system and apply repeatedly steps 2 and 3 (where "variables" there should be read as "nodes") until there is no more change. The set $L_{<1}$ of deficient nodes in step 2 are the type L nodes that correspond to deficient polynomials $p_{ij}(x)$ in $P(x)$. The set $L_{>0}$ of leaky nodes in step 3 are the type L nodes that correspond to polynomials $p_{ij}(x)$ with a positive constant term. When we remove type L nodes (variables) in the algorithm, we remove the corresponding polynomials from the functions $P_i(x)$ where they appear. At the end of Phase 1, we get a reduced maxPPS with the property that $p_{ij}(1) = 1$ for every remaining polynomial. With a more careful analysis of Phase 1, it is easy to show that it converges in at most $n + 1$ iterations where n is the number of type M nodes in G_r, i.e. the number of variables in the given maxPPS. To see this, suppose that an iteration of steps 2-3 does not remove any type M node. Since the type Q nodes have edges only to type M nodes, and type L nodes have edges only to type Q nodes, step 3 does not remove any type Q or L nodes either, and the algorithm terminates. Thus, Phase 1 takes $O(nL)$ time, where L is the encoding size of the given maxPPS.

Then we apply the Phase 2 algorithm using the definition of the operator Q for systems in general form, given at the end of the subsection on minPPS. The LP in Step 1 has at most n variables, $m + 2n$ constraints (where $m = \sum_i m_i$ is the total number of polynomials), and encoding size $L = |P|$. Steps 4-6 use the refined graph of the remaining system, where we consider each type L node corresponding to a polynomial $p_{ij}(x)$ as having an associated value $\hat{p}_{ij}(v)$, where v is the optimal solution computed in step 1. Thus, in step 5, we can use for a type M node x_i only the edges $x_i \to p_{ij}$ such that $v_i = \hat{p}_{ij}(v)$. As in the case of Phase 1, it can be shown that the inner

[7]This is standard: Suppose that q is an optimal solution to an LP $\max\{cx|Ax = b, x \geq 0\}$, such that the set of columns A_i of A corresponding to the index set $S = \{i|q_i > 0\}$ is linearly dependent, i.e., $\sum_{i\in S} \alpha_i A_i = 0$ for some α_i not all 0. We can assume wlog that some $\alpha_i > 0$, let $t = \min\{q_i/\alpha_i | i \in S, \alpha_i > 0\}$, and define the point q' where $q'_i = q_i - t\alpha_i$ for $i \in S$, and $q'_i = q_i$ for $i \notin S$. Then q' is also an optimal feasible solution with (at least) one more 0 coordinate. Repeating this process leads to an optimal bfs.

loop of steps 4-6 will be executed at most $O(n)$ times. The outer loop 1-8 is also executed at most n times. Thus, the whole algorithm takes time $O(n^2L + nT_{LP}(O(n), O(m), O(L)))$. The proof of correctness is along the same lines as the SNF case.

The algorithm for computing an optimal policy for the variables with value 1 in the LFP is again similar to that of the SNF case. The dual LPs now have at most $m + n \leq 2m$ variables, and n constraints in addition to non-negativity. Thus, we have the following.

Theorem A.19. *We can classify qualitatively the variables of a maxPPS in general form with n variables, m polynomials and encoding size L in time $O(n^2L + nT_{LP}(O(n), O(m), O(L)))$. We can compute an optimal policy for the variables that have value 1 in the LFP in time $O(n^2L + nT_{LP}(O(m), O(m), O(L)))$.*

Remark: Observe that for both maxPPSs and minPPSs, the P-time algorithms given in this section, and those given in [19], for deciding whether $q_i^* = 1$ involve linear programming. Thus they do not run in strongly polynomial time. We remark that a strongly polynomial time algorithm for these problems would already imply a substantial advance in the *quantitative* analysis of finite-state MDPs. Specifically, in [19] (Section 7), we showed that there is a simple P-time reduction from the quantitative problem for deciding whether the value of a finite-state simple stochastic game (SSG) is $\geq 1/2$ (known as Condon's problem), to the *qualitative* termination decision problem of deciding whether $q_i^* = 1$ for 1-RSSGs (equivalently, for BSSGs). In fact, an examination of that reduction shows that it also implies a simple *algebraic* (and hence strongly) P-time reduction from the problem of deciding whether a finite-state MDP with *reachability* objective has optimal probability value $\geq 1/2$, to the problem of deciding whether in a given BMDP we have optimal extinction probability $q_i^* = 1$. The only known provably P-time algorithms for computing the optimal value, and solving the quantitative reachability decision problem, for finite-state MDPs (as well as finite-state MDPs with other objectives), involve linear programming (see, e.g., [29, 7]), and are thus not strongly polynomial time.[8] Hence, a strongly polynomial time algorithm for the qualitative $q_i^* = 1$ decision problem for BMDPs would already entail a significant advance for the analysis of finite-state MDPs.

B Omitted material from Sections 2 - 4.

B.1 Omitted material from Section 2.

Proof of Proposition 2.7
We can easily convert, in P-time, any max/minPPS into SNF form, using the following procedure.

- For each equation $x_i = P_i(x) = \max\{p_1(x), \ldots, p_m(x)\}$, for each $p_j(x)$ on the right-hand-side that is not a variable, add a new variable x_k, replace $p_j(x)$ with x_k in $P_i(x)$, and add the new equation $x_k = p_j(x)$. Do similarly if $P_i(x) = \min\{p_1(x), \ldots, p_m(x)\}$.
- If $P_i(x) = \max\{x_{j_1}, ..., x_{j_m}\}$ with $m > 2$, then add $m - 2$ new variables $x_{i_1}, \ldots, x_{i_{m-2}}$, set $P_i(x) = \max\{x_{j_1}, x_{i_1}\}$, and add the equations $x_{i_1} = \max\{x_{j_2}, x_{i_2}\}$, $x_{i_2} = \max\{x_{j_3}, x_{i_3}\}$, ..., $x_{i_{m-2}} = \max\{x_{j_{m-1}}, x_{j_m}\}$. Do similarly if $P_i(x) = \min\{x_{j_1}, ..., x_{j_m}\}$ with $m > 2$.

[8] Strongly-polynomial time algorithms are known for computing the value of finite-state MDPs with restricted objectives, e.g., with a discounted reward objective and a fixed discount $\beta < 1$ (see [32]), or for so-called deterministic MDPs ([28]). However, the reachability objective and other objectives, like the richer mean-payoff objective, are not known to be (algebraically) strongly polynomial time reducible to these objectives.

- For each equation $x_i = P_i(x) = \sum_{j=1}^{m} p_j x^{\alpha_j}$, where $P_i(x)$ is a probabilistic polynomial that is not just a constant or a single monomial, replace every monomial x^{α_j} on the right-hand-side that is not a single variable by a new variable x_{i_j} and add the equation $x_{i_j} = x^{\alpha_j}$.
- For each variable x_i that occurs in some polynomial with exponent higher than 1, introduce new variables $x_{i_1}, \ldots, x_{i_k}$ where k is the logarithm of the highest exponent of x_i that occurs in $P(x)$, and add equations $x_{i_1} = x_i^2$, $x_{i_2} = x_{i_1}^2$, ..., $x_{i_k} = x_{i_{k-1}}^2$. For every occurrence of a higher power x_i^l, $l > 1$, of x_i in $P(x)$, if the binary representation of the exponent l is $a_k \ldots a_2 a_1 a_0$, then we replace x_i^l by the product of the variables x_{i_j} such that the corresponding bit a_j is 1, and x_i if $a_0 = 1$. After we perform this replacement for all the higher powers of all the variables, every polynomial of total degree >2 is just a product of variables.
- If a polynomial $P_i(x) = x_{j_1} \cdots x_{j_m}$ in the current system is the product of $m > 2$ variables, then add $m - 2$ new variables $x_{i_1}, \ldots, x_{i_{m-2}}$, set $P_i(x) = x_{j_1} x_{i_1}$, and add the equations $x_{i_1} = x_{j_2} x_{i_2}$, $x_{i_2} = x_{j_3} x_{i_3}$, ..., $x_{i_{m-2}} = x_{j_{m-1}} x_{j_m}$.

Now all equations are of the form L, Q, or M.

The above procedure allows us to convert any max/minPPS into one in SNF form by introducing $O(|P|)$ new variables and blowing up the size of P by a constant factor $O(1)$. Furthermore, there is an obvious (and easy to compute) bijection between policies for the resulting SNF form max/minPPS and the original max/minPPS. □

We will give in the rest of this section some lemmas on PPS which we will need in this paper. As usual, we always assume, w.l.o.g., that PPS are in SNF form.

Lemma B.1. *(cf. Lemma 3.3 of [13]) For any PPS $x = P(x)$ in SNF with n variables, and any pair of vectors $a, b \in \mathbb{R}^n$, $P(a) - P(b) = P'(\frac{a+b}{2})(a - b)$.*

Lemma B.2. *Given a PPS, $x = P(x)$, with LFP $q^* > 0$, if $0 \leq y \leq q^*$, and if $(I - P'(y))^{-1}$ exists and is non-negative (in which case clearly $\mathcal{N}(y)$ is defined), then $\mathcal{N}(y) \leq q^*$ holds.*[9]

Proof. In Lemma 3.4 of [13] it was established that when $(I - P'(y))$ is non-singular, i.e., $(I - P'(y))^{-1}$ is defined, and thus $\mathcal{N}(y)$ is defined, then

$$q^* - \mathcal{N}(y) = (I - P'(y))^{-1} \frac{P'(q^*) - P'(y)}{2}(q^* - y) \tag{13}$$

Now, since all polynomials in $P(x)$ have non-negative coefficients, it follows that the Jacobian $P'(x)$ is monotone in x, and thus since $y \leq q^*$, we have that $P'(q^*) \geq P'(y)$. Thus $(P'(q^*) - P'(y)) \geq 0$, and by assumption $(q^* - y) \geq 0$. Thus, by the assumption that $(I - P'(y))^{-1} \geq 0$, we have by equation (13) that $q^* - \mathcal{N}(y) \geq 0$, i.e., that $q^* \geq \mathcal{N}(y)$. □

We also will need the following, which is a less immediate consequence of results in [13]. Recall that for a square matrix A, $\rho(A)$ denotes its spectral radius.

Lemma B.3. *Given a PPS, $x = P(x)$, with LFP $q^* > 0$, if $0 \leq y \leq q^*$, and $y < 1$, then $\rho(P'(y)) < 1$, and $(I - P'(y))^{-1}$ exists and is non-negative.*

[9] Note that the Lemma does not claim that $\mathcal{N}(y) \geq 0$ holds. Indeed, it may not.

The proof of this lemma is more involved. We first recall several closely related results established in our previous papers. Recall that a PPS, $x = P(x)$, is called *strongly connected*, if its variable dependency graph H is strongly connected.

Lemma B.4. *(Lemma 6.5 of [18])*[10] *Let $x = P(x)$ be a strongly connected PPS, in n variables, with LFP $q^* > 0$. For any vector $0 \leq y < q^*$, $\rho(P'(y)) < 1$, and thus $(I - P'(y))^{-1}$ exists and is nonnegative.*

Lemma B.5. *(Theorem 3.7 of [13]) For any PPS, $x = P(x)$, in SNF form, which has LFP $0 < q^* < 1$, for all $0 \leq y \leq q^*$, $\rho(P'(y)) < 1$ and $(I - P'(y))^{-1}$ exists and is nonnegative.*

Proof of Lemma B.3. Consider a PPS, $x = P(x)$, with LFP $q^* > 0$, and a vector $0 \leq y \leq q^*$, such that $y < 1$. Note that all we need to establish is that $\rho(P'(y)) < 1$, because it then follows by standard facts (see, e.g., [23]) that $(I - P'(y))^{-1}$ exists and is equal to $\sum_{i=0}^{\infty}(P'(y))^i \geq 0$.

Let us first show that if $x = P(x)$ is strongly connected, then $\rho(P'(y)) < 1$. To see this, note that if $x = P(x)$ is strongly connected, then every variable depends on every other, and thus if there exists any $i \in \{1, \ldots, n\}$ such that $q_i^* < 1$, then it must be the case that for all $j \in \{1, \ldots, n\}$, we have $q_j^* < 1$. Thus, either $q^* = 1$, or else $0 < q^* < 1$. If $q^* = 1$, then since $y < 1$, we have $y < q^*$, and thus, by Lemma B.4, we have $\rho(P'(y)) < 1$. If, on the other hand, $0 < q^* < 1$, then since $0 \leq y \leq q^*$, by Lemma B.5, we have $\rho(P'(y)) < 1$.

Next, consider an arbitrary PPS, $x = P(x)$, that is not necessarily strongly connected. Recall the variable dependency graph H of $x = P(x)$. We can partition the variables into sets $S_1, \ldots, S_k$ which form the SCCs of H. Consider the DAG, D, of SCCs, whose nodes are the sets S_i, and for which there is an edge from S_i to S_j iff in the dependency graph H there is a node $i' \in S_i$ with an edge to a node in $j' \in S_j$.

Consider the matrix $P'(y)$. Our aim is to show that $\rho(P'(y)) < 1$. Since we assume $q^* > 0$, $0 \leq y \leq q^*$, and $y < 1$, it clearly suffices to show that $\rho(P'(y)) < 1$ holds in the case where we additionally insist that $y > 0$, because then for any other z such that $0 \leq z \leq y$, we would have $\rho(P'(z)) \leq \rho(P'(y)) < 1$.

So, assuming also that $y > 0$, consider the $n \times n$-matrix $P'(y)$. To keep notation clean, we let $A := P'(y))$. For the $n \times n$ matrix A, we can consider its underlying *dependency* graph, $H = (\{1, \ldots, n\}, E_H)$, whose nodes are $\{1, \ldots, n\}$, and where there is an edge from i to j iff $A_{i,j} > 0$. Notice however that, since $y > 0$, this graph is precisely the same graph as the dependency graph H of $x = P(x)$, and thus it has the same SCCs, and the same DAG of SCCs, D. Let us sort the SCCs, so that we can assume $S_1, \ldots, S_k$ are topologically sorted with respect to the partial ordering defined by the DAG D. In other words, for any variable indices $i \in S_a$ and $j \in S_b$ if $(i, j) \in E_H$, then $a \leq b$.

Let $S \subseteq \{1, \ldots, n\}$ be any non-empty subset of indices, and let $A[S]$ denote the principle submatrix of A defined by indices in S. It is a well known fact that $0 \leq \rho(A[S]) \leq \rho(A)$. (See, e.g, Corollary 8.1.20 of [23].)

Since $A \geq 0$, $\rho(A)$ is an eigenvalue of A, and has an associated non-negative eigenvector $v \geq 0$, $v \neq 0$ (again see, e.g., Chapter 8 of [23]). In other words,

$$Av = \rho(A)v$$

[10] Lemma 6.5 of [18] is actually a more general result, relating to strongly connected MPSs that arise from more general RMCs.

Firstly, if $\rho(A) = 0$, then we are of course trivially done. So we can assume w.l.o.g. that $\rho(A) > 0$. Now, if $v_i > 0$, then for every j such that $(j, i) \in E_H$, we have $(Av)_j > 0$, and thus since $(Av)_j = \rho(A)v_j$, we have $v_j > 0$. Hence, repeating this argument, if $v_i > 0$ then for every j that has a path to i in the dependency graph H, we have $v_j > 0$.

Since $v \neq 0$, it must be the case that there is exists some SCC, S_c, of H such that for every variable index $i \in S_c$, $v_i > 0$, and furthermore, such that c is the maximum index for such an SCC in the topologically sorted list $S_1, \ldots, S_k$, i.e., such that for all $d > c$, and for all $j \in S_d$, we have $v_j = 0$.

First, let us note that it must be the case that S_c is a *non-trivial* SCC. Specifically, let us call an SCC, S_r of H *trivial* if $S_r = \{i\}$ consists of only a single variable index, i, and furthermore, such that $\mathbf{0} = (A)_i = (P'(y))_i$, i.e., that row i of the matrix A is all zero. This can not be the case for S_c, because for any variable $i \in S_c$, we have $v_i > 0$, and thus $(Av)_i = \rho(A)v_i > 0$.

Let us consider the principal submatrix $A[S_c]$ of A. We claim that $\rho(A[S_c]) = \rho(A)$. To see why this is the case, note that $Av = \rho(A)v$, and for every $i \in S_c$, we have $(Av)_i = \sum_j a_{i,j}v_j = \rho(A)v_i$. But $v_j = 0$ for every $j \in S_d$ such that $d > c$, and furthermore $a_{i,j} = 0$ for every $j \in S_{d'}$ such that $d' < c$.

Thus, if we let v_{S_c} denote the subvector of v corresponding to the indices in S_c, then we have just established that $A[S_c]v_{S_c} = \rho(A)v_{S_c}$, and thus that $\rho(A[S_c]) \geq \rho(A)$. But since $A[S_c]$ is a principal submatrix of A, we also know easily (see, e.g, Corollary 8.1.20 of [23]), that $\rho(A[S_c]) \leq \rho(A)$, so $\rho(A[S_c]) = \rho(A)$.

We are almost done. Given the original PPS, $x = P(x)$, for any subset $S \subseteq \{1, \ldots, n\}$ of variable indices, let $x_S = P_S(x_S, x_{D_S})$ denote the subsystem of $x = P(x)$ associated with the vector x_S of variables in set S, where x_{D_S} denotes the variables not in S.

Now, note that $x_{S_c} = P_{S_c}(x_{S_c}, y_{D_{S_c}})$ is itself a PPS. Furthermore, it is a *strongly connected* PPS, precisely because S_c is a strongly connected component of the dependency graph H, and because $y > 0$. Moreover, the Jacobian matrix of $P_{S_c}(x_{S_c}, y_{D_{S_c}}))$, evaluated at y_{S_c}, which we denote by $P'_{S_c}(y)$, is precisely the principal submatrix $A[S_c]$ of A. Since $x_{S_c} = P_{S_c}(x_{S_c}, y_{D_{S_c}})$ is a strongly connected PPS, we have already argued that it must be the case that $\rho(P'_{S_c}(y)) < 1$. Thus since $P'_{S_c}(y) = A[S_c]$, we have $\rho(A[S_c]) = \rho(A) < 1$. This completes the proof. □

B.2 Omitted material from Section 3.

Proof of Lemma 3.6.
We need to show that the Jacobian $(P^y)'(x)$ of $P^y(x)$, evaluated anywhere, is equal to $P'(y)$. If $x_i = P_i(x)$ is not of form Q, then, for any $x \in \mathbb{R}^n$, $P_i(x) = P_i^y(x)$. So for any x_j, $\frac{\partial P_i^y(x)}{\partial x_j} = \frac{\partial P_i(x)}{\partial x_j}$. Otherwise, $x_i = P_i(x)$ has form Q, that is $P_i(x) = x_j x_k$ for some variables x_j,x_k. Then $P_i^y(x) = y_j x_k + x_j y_k - y_j y_k$. In this case $\frac{\partial P_i^y(x)}{\partial x_j} = y_k$ and $\frac{\partial P_i^y(x)}{\partial x_k} = y_j$. But when $x = y$, $\frac{\partial P_i(x)}{\partial x_j} = y_k$ and $\frac{\partial P_i(x)}{\partial x_k} = y_j$. Furthermore, clearly for any x_l, with $l \neq j$ and $l \neq k$, $\frac{\partial P_i(x)}{\partial x_l} = 0$ and $\frac{\partial P_i^y(x)}{\partial x_l} = 0$. We have thus established that $(P^y)'(x) = P'(y)$ for any $x \in \mathbb{R}^n$. □

Proof of Lemma 3.7.
Firstly, note that $P^y(x) = P^y(y) + (P^y)'(x)(x - y)$, since the functions $P_i^y(x)$ are all linear in x. Next, observe that $P_i(y) = P_i^y(y)$, for all i, and thus that $P(y) = P^y(y)$. Thus, to show that $P^y(x) = P^y(y) + P'(y)(x - y) = P(y) + P'(y)(x - y)$, all we need to show is that the Jacobian $(P^y)'(x)$ of $P^y(x)$, evaluated anywhere, is equal to $P'(y)$. But this was established in Lemma 3.6.

□

Proof of Lemma 3.8.
(i): We define:

$$a = y + (I - P'_\sigma(y))^{-1}(P_\sigma(y) - y) \equiv \mathcal{N}_\sigma(y)$$

Then we can re-arrange this expression, reversibly, yielding:

$$\begin{aligned} a = y + (I - P'_\sigma(y))^{-1}(P_\sigma(y) - y) &\Leftrightarrow P_\sigma(y) - y - (I - P'_\sigma(y))(a - y) = 0 \\ &\Leftrightarrow P_\sigma(y) + P'_\sigma(y)(a - y) = a \\ &\Leftrightarrow P^y_\sigma(a) = a \qquad \text{(by Lemma 3.7)} \end{aligned}$$

Uniqueness follows from the reversibility of these transformations.

(ii): Firstly, we shall observe that the result of applying Newton's method to solve $x = P^y_\sigma(x)$ with any initial point x gives us $\mathcal{N}_\sigma(y) = a$ in a single iteration. Recalling from Lemma 3.6 that the following equality holds between the Jacobians: $(P^y_\sigma)'(x) = P'_\sigma(y)$, one iteration of Newton's method applied to $x = P^y_\sigma(x)$ can be equivalently defined as:

$$\begin{aligned} x + (I - P'_\sigma(y))^{-1}(P^y_\sigma(x) - x) &= x + (I - P'_\sigma(y))^{-1}(P_\sigma(y) + P'_\sigma(y)(x - y) - x) \\ &= (I - P'_\sigma(y))^{-1}(x - P'_\sigma(y)x + P_\sigma(y) + P'_\sigma(y)(x - y) - x) \\ &= (I - P'_\sigma(y))^{-1}(P_\sigma(y) - P'_\sigma(y)y) \\ &= (I - P'_\sigma(y))^{-1}((I - P'_\sigma(y))y + P_\sigma(y) - y) \\ &= y + (I - P'_\sigma(y))^{-1}(P_\sigma(y) - y) \\ &= \mathcal{N}_\sigma(y). \end{aligned}$$

We thus have $\mathcal{N}_\sigma(y) = x + (I - P'_\sigma(y))^{-1}(P^y_\sigma(x) - x)$. By assumption, $(I - P'_\sigma(y))^{-1}$ is a non-negative matrix. So if $P^y_\sigma(x) - x \geq 0$ then $\mathcal{N}_\sigma(y) \geq x$, whereas if $P^y_\sigma(x) - x \leq 0$ then $\mathcal{N}_\sigma(y) \leq x$. □

Proof of Lemma 3.18.
Firstly, we show that $P'(y)(1 - y) \leq (1 - y)$. Clearly, for any PPS, $P(1) \leq 1$. Note that since by assumption $y \leq P(y)$, we have $(1 - y) \geq (1 - P(y)) \geq (P(1) - P(y))$. Then by Lemma B.1 (Lemma 3.3 of [13]):

$$\begin{aligned} (1 - y) \geq P(1) - P(y) &= P'(\frac{1 + y}{2})(1 - y) &(14) \\ &\geq P'(y)(1 - y) &(15) \end{aligned}$$

Again by Lemma B.1: $P(y) - P(x) = \frac{1}{2}(P'(x) + P'(y))(y - x)$, and thus:

$$P(x) = P(y) - \frac{1}{2}(P'(x) + P'(y))(y - x) \tag{16}$$

Thus:

$$\begin{aligned}
y - \mathcal{N}(x) &= y - x - (I - P'(x))^{-1}(P(x) - x) \\
&= y - x - (I - P'(x))^{-1}(P(y) - x - \frac{1}{2}(P'(x) + P'(y))(y - x)) \quad \text{(by (16))} \\
&\leq y - x - (I - P'(x))^{-1}(y - x - \frac{1}{2}(P'(x) + P'(y))(y - x)) \\
&= (y - x) - (I - P'(x))^{-1}((y - x) - \frac{1}{2}(P'(x) + P'(y))(y - x)) \\
&= (I - (I - P'(x))^{-1}(I - \frac{1}{2}(P'(x) + P'(y))))(y - x) \\
&= ((I - P'(x))^{-1}(I - P'(x)) - (I - P'(x))^{-1}(I - \frac{1}{2}(P'(x) + P'(y))))(y - x) \\
&= (I - P'(x))^{-1}(I - P'(x) - (I - \frac{1}{2}(P'(x) + P'(y))))(y - x) \\
&= (I - P'(x))^{-1}(-P'(x) + \frac{1}{2}(P'(x) + P'(y)))(y - x) \\
&= (I - P'(x))^{-1}\frac{1}{2}(P'(y) - P'(x))(y - x) \\
&\leq \frac{\lambda}{2}(I - P'(x))^{-1}(P'(y) - P'(x))(1 - y) \quad \text{(by (4), and because } (P'(y) - P'(x)) \geq 0) \\
&\leq \frac{\lambda}{2}(I - P'(x))^{-1}(I - P'(x))(1 - y) \quad \text{(because by (15), } P'(y)(\mathbf{1} - y) \leq (\mathbf{1} - y)) \\
&= \frac{\lambda}{2}(1 - y)
\end{aligned}$$

□

B.3 Omitted material from Section 4

Proof of Lemma 4.2.
Lemma B.1 tells us that for any PPS, $x = P(x)$, (assumed to be in SNF form), and any pair of vectors $a, b \in \mathbb{R}^n$, we have $P(a) - P(b) = P'((a + b)/2)(a - b)$. Applying this lemma with $a = q^*$ and $b = y$, we have that

$$q^* - P(y) = P'((1/2)(q^* + y))(q^* - y)$$

Subtracting both sides from $q^* - y$, we have that:

$$P(y) - y = (I - P'((1/2)(q^* + y)))(q^* - y) \tag{17}$$

Multiplying both sides of equation (17) by $(I - P'((1/2)(q^* + y)))^{-1}$, we obtain:

$$q^* - y = (I - P'(1/2(q^* + y)))^{-1}(P(y) - y)$$

as required. □

Proof of Lemma 4.3.
We show first the lemma for PPS, and then extend it to max/minPPS. Suppose that $x = P(x)$ is a PPS (recall, in SNF). By Lemma B.1, we have that $q^* - P(y) = P'(\frac{1}{2}(y + q^*))(q^* - y)$. Since

$\frac{1}{2}(y+q^*) \leq 1$, $\|P'(\frac{1}{2}(y+q^*))\|_\infty \leq 2$: If the ith row has $x_i = P_i(x)$ of type L then $\sum_{j=1}^n |p_{i,j}| \leq 1$ and if $x_i = P_i(x)$ has type Q, i.e., $P_i(x) = x_j x_k$ for some j,k, then $\sum_{j=1}^n |\frac{\partial P_i(x)}{\partial x_j}(\frac{1}{2}(y+q^*))| = \frac{1}{2}(y_j+q_j^*)+\frac{1}{2}(y_k+q_k^*) \leq 2$. So we have that $\|q^*-P(y)\|_\infty \leq \|P'(\frac{1}{2}(y+q^*))\|_\infty \|q^*-y\|_\infty \leq 2\|q^*-y\|_\infty$.

Since $y \leq q^*$, we know also that $P(y) \leq q^* = P(q^*)$ since $P(x)$ is monotone. If $(P(y))_i \leq y_i$, then $y_i - P(y)_i \leq q_i^* - P(y)_i \leq \|q^*-P(y)\|_\infty \leq 2\|q^*-y\|_\infty$. If $P_i(y) \geq y_i$, $P_i(y) - y_i \leq q_i^* - y_i \leq \|q^*-y\|_\infty$. So $\|P(y)-y\|_\infty \leq 2\|q^*-y\|_\infty$ as required.

Suppose now that $x = P(x)$ is a max/minPPS. Then it has some optimal policy, τ, and from the above, $\|P_\tau(y)-y\|_\infty \leq 2\|q^*-y\|_\infty$. For type L and type Q variable x_i, we have $(P_\tau)_i(y) = P_i(y)$. It thus only remains to show that $|P_i(y)-y_i| \leq 2\|q^*-y\|_\infty$ when x_i is of type M.

If $P_i(y) \geq y_i$, then this follows easily: as before we have that $P_i(y) - y_i \leq q_i^* - y_i \leq \|q^*-y\|_\infty$. Suppose that instead we have $P_i(y) \leq y_i$. Then we consider the two cases (min and max) separately to bound $|P_i(y)-y_i| = y_i - P_i(y)$:

Suppose $x = P(x)$ is a minPPS, and that $P_i(x) = \min\{x_j, x_k\}$. Since $q^* = P(q^*)$, we have:

$$0 \leq y_i - P_i(y) \leq q_i^* - P_i(y) = \min\{q_j^*, q_k^*\} - P_i(y) \tag{18}$$

We can assume, w.l.o.g., that $P_i(y) \equiv \min\{y_j, y_k\} = y_j$. (The case where $P_i(y) = y_k$ is entirely analogous.) Then, by (18), we have:

$$0 \leq y_i - P(y)_i \leq \min\{q_j^*, q_k^*\} - y_j \leq q_j^* - y_j \leq \|q^*-y\|_\infty$$

Suppose now that $x = P(x)$ is a maxPPS, and that $P_i(x) \equiv \max\{x_j, x_k\}$. Again, we are already assuming that $P_i(y) \leq y_i$. Since $q^* = P(q^*)$, we have:

$$0 \leq y_i - P_i(y) \leq q_i^* - P_i(y) = P_i(q^*) - \max\{y_j, y_k\} \tag{19}$$

We can assume, w.l.o.g., that $P_i(q^*) \equiv \max\{q_j^*, q_k^*\} = q_j^*$. (Again, the case when $P_i(q^*) = q_k^*$ is entirely analogous.) Then, by (19), we have:

$$0 \leq y_i - P_i(y) \leq q_j^* - \max\{y_j, y_k\} \leq q_j^* - y_j \leq \|q^*-y\|_\infty$$

This completes the proof of the Lemma for all max/minPPSs. □

Proof of Lemma 4.4.
We will apply Lemma 4.2 to the PPS $x = P_\sigma(x)$ (which has LFP q_σ^*), with q in place of y; for this we need to show that $(I - P'_\sigma(\frac{1}{2}(q_\sigma^* + q)))^{-1}$ exists. Note that since $0 \leq q \leq q^*$, we have $0 \leq P'_\sigma(\frac{1}{2}(q_\sigma^* + q)) \leq P'_\sigma(\frac{1}{2}(q_\sigma^* + q^*))$, and thus $0 \leq \rho(P'_\sigma(\frac{1}{2}(q_\sigma^* + q))) \leq \rho(P'_\sigma(\frac{1}{2}(q_\sigma^* + q^*)) < 1$. Therefore, $(I - (P'_\sigma(\frac{1}{2}(q_\sigma^* + q)))^{-1}$ also exists and is non-negative. Lemma 4.2 gives $q_\sigma^* - q = (I - P'_\sigma(\frac{1}{2}(q_\sigma^* + q)))^{-1}(P_\sigma(q) - q)$. Taking norms, we obtain the following inequality:

$$\|q_\sigma^* - q\|_\infty \leq \|(I - P'_\sigma(\frac{1}{2}(q_\sigma^* + q)))^{-1}\|_\infty \|P_\sigma(q) - q\|_\infty \tag{20}$$

Using the fact that $P_\sigma(q) = P(q)$ and Lemma 4.3, we have:

$$\begin{aligned}
\|q^* - q^*_\sigma\|_\infty &\leq \|q^* - q\|_\infty + \|q^*_\sigma - q\|_\infty \\
&\leq \|q^* - q\|_\infty + \|(I - P'_\sigma(\frac{1}{2}(q^*_\sigma + q)))^{-1}\|_\infty \|P_\sigma(q) - q\|_\infty \\
&= \|q^* - q\|_\infty + \|(I - P'_\sigma(\frac{1}{2}(q^*_\sigma + q)))^{-1}\|_\infty \|P(q) - q\|_\infty \\
&\leq \|q^* - q\|_\infty + \|(I - P'_\sigma(\frac{1}{2}(q^*_\sigma + q)))^{-1}\|_\infty 2\|q^* - q\|_\infty \\
&= (2\|(I - P'_\sigma(\frac{1}{2}(q^*_\sigma + q)))^{-1}\|_\infty + 1)\|q^* - q\|_\infty \\
&\leq (2\|(I - P'_\sigma(\frac{1}{2}(q^*_\sigma + q^*)))^{-1}\|_\infty + 1)\|q^* - q\|_\infty
\end{aligned}$$

The last inequality follows because $q \leq q^*$, and

$$0 \leq (I - P'_\sigma(q^*_\sigma + q))^{-1} = \sum_{i=0}^{\infty} (P'_\sigma(q^*_\sigma + q))^i \leq \sum_{i=0}^{\infty} (P'_\sigma(q^*_\sigma + q^*))^i = (I - P'_\sigma(q^*_\sigma + q^*))^{-1}.$$

□

Proof of Lemma 4.9.
(i) $\implies$ (ii): From [18], $P^k(0) \to q^*$ as $k \to \infty$. It follows that if $(P^k(0))_i = 0$ for all k, then $q^*_i = 0$.
(ii) $\implies$ (iii): From [18], $P^k(0)$ is monotonically non-decreasing in k, i.e., if $m \geq l > 0$ then $P^m(0) \geq P^l(0)$. Thus, if $(P^k(0))_i > 0$ for some $k \leq n$, then $(P^n(0))_i > 0$.

Whether $P_i(x) > 0$ depends only on whether each $x_j > 0$ or not and not on the value of x_j. So, for any k, whether $(P^{k+1}(0))_i > 0$ depends only on the set $S_k = \{x_j | (P^k(0))_j > 0\}$. Since $P^{k+1}(0) \geq P^k(0)$, we have $S_{k+1} \supseteq S_k$. If ever we have that $S_{k+1} = S_k$, then for any j, $(P^{k+2}(0))_j > 0$ whenever $(P^{k+1}(0))_j > 0$ so $S_{k+2} = S_{k+1} = S_k$. $S_{k+1} \supset S_k$ can only occur for n values of k as there are only n variables to add. Consequently $S_{n+1} = S_n$ and so $S_m = S_n$ whenever $m > n$. So if we have a $k > n$ with $(P^k(0))_i > 0$, then $(P^n(0))_i > 0$.
(iii) $\implies$ (i): By monotonicity and an easy induction, $q^* \geq P^k(0)$ for all $k > 0$. In particular $q^* \geq P^n(0)$. So $q^*_i \geq (P^n(0))_i > 0$. □